Thesis for the degree of Doctor of Philosophy

Energy and Route Optimization of Moving Devices

Sarmad Riazi

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2020

Energy and Route Optimization of Moving Devices
Sarmad Riazi
ISBN 978-91-7905-286-7

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 4753
ISSN 0346-718X

Department of Electrical Engineering
Chalmers University of Technology
SE–412 96 Göteborg, Sweden
Telephone + 46 (0) 31 – 772 1000

Cover: *Picea abies* forest, Korpimäcki reserve, Sweden. Forests are the prime example of sustainable systems. The idea of the cover image was inspired by a conversation with Ian Green of Greenheart TreeWalk, Vacouver, Canada. Photograph by unknown Wikipedia user, distributed under a CC BY-SA 3.0 license.

Typeset by the author using LaTeX.

Printed by Chalmers Reproservice
Göteborg, Sweden 2020

to my family, and to those who care

Abstract

This thesis highlights our efforts in energy and route optimization of moving devices. We have focused on three categories of such devices; industrial robots in a multi-robot environment, generic vehicles in a vehicle routing problem (VRP) context, automated guided vehicles (AGVs) in a large-scale flexible manufacturing system (FMS).

In the first category, the aim is to develop a non-intrusive energy optimization technique, based on a given set of paths and sequences of operations, such that the original cycle time is not exceeded. We develop an optimization procedure based on a mathematical programming model that aims to minimize the energy consumption and peak power. Our technique has several advantages. It is non-intrusive, i.e. it requires limited changes in the robot program and can be implemented easily. Moreover, it is model-free, in the sense that no particular, and perhaps secret, parameter or dynamic model is required. Furthermore, the optimization can be done offline, within seconds using a generic solver. Through careful experiments, we have shown that it is possible to reduce energy and peak-power up to about 30% and 50% respectively.

The second category of moving devices comprises of generic vehicles in a VRP context. We have developed a hybrid optimization approach that integrates a distributed algorithm based on a gossip protocol with a column generation (CG) algorithm, which manages to solve the tested problems faster than the CG algorithm alone. The algorithm is developed for a VRP variation including time windows (VRPTW), which is meant to model the task of scheduling and routing of caregivers in the context of home healthcare scheduling and routing problems (HHCSRP). Moreover, the developed algorithm can easily be parallelized to further increase its efficiency.

The last category deals with AGVs. The choice of AGVs was not arbitrary; by design, we decided to transfer our knowledge of energy optimization and routing algorithms to a class of moving devices in which both techniques are of interest. Initially, we improve an existing method of conflict-free AGV scheduling and routing, such that the new algorithm can manage larger problems. A heuristic version of the algorithm manages to solve the problem instances in a reasonable amount of time. Later, we develop strategies to reduce the energy consumption. The study is carried out using an AGV system installed at Volvo Cars. The results are promising; (1) the algorithm reduces performance measures such as makespan up to 50%, while reducing the total traveled distance of the vehicles about 14%, leading to an energy saving of roughly 14%, compared to the results obtained from the original traffic controller. (2) It is possible to reduce the cruise velocities such that more energy is saved, up to 20%, while the new makespan remains better than the original one.

Keywords: Energy optimization, routing, scheduling, industrial robots, vehicle routing problems, automated guided vehicles.

Acknowledgments

I will be honest with you. Every time I pick up a new thesis or book, the very first part I read is the acknowledgment section. If I happen to know the author personally, my eyes would frantically search for *Sarmad* in the section while my heart is pounding in my chest. Was the author kind enough to include me? After all, we have been friends, haven't we? So, if you are feeling the same now, I do hope you will find your name below.

I have been thinking for several years about how to write this section. I have been tempted to have a very short text and be done with it, but that's not how I work. So, get ready for what is probably the longest acknowledgment section in a PhD thesis.

Bengt Lennartson On a sunny day of late August 2011, and on the very first moments of sitting in a Swedish classroom, overwhelmed with anticipation and excitement, you approached me with a big smile and greeted me in your DES course. And that was the beginning of an almost decade-long, I dare say, *friendship*, which I hope will last for the years to come. You sure did know how to recruit, might I say, *good*, students. Yes, you have been my supervisor for all these years, and I have had my fare share moments. The bottom-line is that, I am very grateful for the many opportunities you provided me with, whether or not I took them. I am especially thankful for recommending me all the way for the position at University West, even though I chose to take another path, for the moment. By the way, I promise you, one day, not so far away, we will sit and have a full conversation in Swedish. Please don't give up on me, as you never have. *Eller hur?*

Martin Fabian You are hands down the nicest boss anyone could wish for. Other than that, I had the privileged of begin your teaching assistant (TA) for five sweet years, which lead to my *best TA* award, which was possible only because of your unconditional non-stop support, which I am forever grateful, and I hope Thomas Eriksson is now reading this! As far as my memory serves, you never said no to any of my requests, because your motto is *a manager's job is to make it possible for his people to work*, and you live up to your standards. To formulate my research method, I used the instructions you sent us a while back. So, thank you for that, and for many more informative emails you sent over the years. Furthermore, you were the first person who introduced me to the world of AGVs in 2017, and something tells me it might have had a *minor* effect on my decision to take the position in AGVE? Thanks for everything Martin. You always *really* cared. Let's keep in touch.

Kristofer Bengtsson On paper, you were my co-supervisor. In reality, you were my mentor, a role model to look up to, and a good friend. As Sabino always says, you are the wise man who does not talk much, but when he says something, it's full of wisdom. I really appreciate you listening to all my complaints, worries, ideas, etc. with receptive ears. I am especially grateful to you for sending me over to Germany as part of the AREUS project. That was one of the sweetest periods during my studies, and I was *not* the same person when I came back. I remember the first day at KUKA was really tough, and I anxiously wrote an email to you full of concerns. Your reply was simple yet effective; *if you don't manage to do everything, it's not the end of the world*. And that was all I needed to hear to calm down, find myself, and do what I had to do. I think you are a very talented manager; not only you guide and inspire your team, but also you know your craft. I remember one guy at KUKA said about you that *this guy really knows SCALA*. It was an amazing experience to work with you in the AREUS, SmoothIT, and the *Regionhabilitering* projects. I hope I can keep collaborating with you, i.e., I hope you want to keep working with me!

Petter Falkman You are a prime example of a person who goes out of his way to help someone when it's absolutely not his responsibility to do so. Many times during my studies, I dropped by your office to ask for help, and you just *listened* and *helped*. It was so easy and pleasant to talk to you, and more importantly, to trust you. I am so grateful to you for introducing me to AGVE and carving out time to take me there for meetings. A funny note; I always came to the *medarbetarsamtal* meeting with you and complained about things, and you just listened and helped, as always. Only in my last year I heard that those meetings were not really meant for me to complain. Instead, medarbetarsamtal was the opportunity for the organization to criticize my performance, which never happened. Thanks for everything.

Balázs Adam Kulcsár I was privileged to be your TA in 2014. You were personally so invested in the course that it also motivated me to do my best. And after the course, you invited your TAs to lunch. How nice of you! Additionally, several times you approached me and asked about my research, which lead to a series of follow-up discussions. It definitely helped me find my way. Thank you for being such a nice person.

The nice people at KUKA I would like to thank Rainer Bischoff, the Head of KUKA Corporate Research, for his support during the AREUS project. Furthermore, special thanks to Andreas Aurnhammer from the path planning team. It has been a very pleasant experience to work with you Andreas. Thank you very much for your continuous support. It was only when I worked with you that I realized there was so much to learn about robots that I had not learned at the university. Moreover, special thanks to Klaus Miller. Klaus, it was so nice to talk to you and I really enjoyed our discussion during the project. Also, big thanks to Matthias Schellhase, Gerhard Werner, and Sebastian Jablonski for their technical support.

The nice people at University West (HV) I would like to thank Bo Svensson, Fredrik Danielsson, and Lennart Malmsköld, for their support during the

PROSAM project. I would also like to thank my good friend and coauthor Emile Glorieux, who was involved in the project at time. It was very nice working with you Emile. I also spent a short while at PTC where I gave an optimization course together with Kenneth Eriksson and Sudha Ramasamy. It was very nice talking to you Kenneth. I will never forget your help and advice. It was also a fun experience to work with you Sudha. I would also like to thank the bosses and managers at HV, in particular, Mikael Ericsson and Per Nylén. My plan was to pursue a research career at HV, but I suddenly got an offer from AGVE, which I could not resist. When I told about it to Mikael, Per, Bo, and Kenneth, they were unbelievably kind and supportive of my decision. Especially Per, you told me that the *doors are not closed, and that I could come back with industrial experience under my belt.* That was a very nice thing to say by a manager who was about to lose a potentially good employee for an unknown amount of time. Thanks to all of you. I hope you let me keep teaching courses at HV every now and then!

The nice people at AGVE To my knowledge, very few doctoral students will have the opportunity to directly use their research results in a relevant company immediately after graduation, on which ground, I am very glad and grateful. Special thanks to my current boss and CTO, Christian Skeppstedt, for offering the position at AGVE. It's the best of both worlds; I can do research on future technologies and products, while collaborating with universities, teaching and supervising students, which is my other passion. Moreover, I thank our CEO, Jakob Dannemark, for providing the resources I needed to carry out my research over the past two years. Big thanks to Thomas Diding, for being such a supportive and nice person. You patiently spent many hours explaining your systems to me, when I was doing research that could lead to nowhere. Yet, you kept supporting me and gave me good ideas to explore. I'm very thankful for all the help, and I hope I will be a nice colleague for you. Finally, special thanks to Jan Svensson and Jonas Martner, for extensive help and support with energy measurements of AGVs.

The good friends and colleagues at Chalmers I have had a lot of fun and many insightful conversations with the following people. *Anton Klintberg*; you got me into stock market for which I am grateful to you, and yes, I fortunately got out before the recent crash. *Amir Hossein Ebrahimi*; you were always like the brother I never had. Thank you for all the useful advice on everything, from programming to life. *Elena Malz*; a few times I had trouble focusing and I discussed it with you. You always reassured me that it happened to everyone. Thanks for the help! *Giuseppe Giordano*; no acknowledgment list is complete without your name on it. Thanks for making the life more enjoyable in the department. Also thanks for giving me an extensive list of nice places to visit in Italy. I should check them out.

Adnan Khan; we shared a lot of conversations and ups and downs. You were the only person who understood me when I thought nobody else did. Thanks man! *Ashfaq Farooqui*; my good friend and former office mate, thanks for all your help and insight regarding life, Linux, books, etc. *Mattias Hovgard*; you are a very nice and very friendly person, but Sabino does not quite get it. I was lucky to take several walks with you on the jugging track near Mossen, and there you taught me many

things about asp och ask och brjörk, etc. That's how I realized what a nice friend you could be. Thanks for all the *urban explorations* we did together. We should do it again. *Sahar Mohajerani*; I was lucky that I asked you about TA-ship. You were the one who told me to become Martin Fabian's TA. I am very grateful for that advice, and for the many useful things you taught me. I especially used your recent advice regarding how to structure the thesis. Thanks for that!

Mona Noori-Hosseini; I asked for your help several times over the years when I felt lost, and you always helped tremendously. Thank you! *Fredrik Hagebring*; you were the one who proposed that we both supervise students in collaboration with Kollmorgen. If it wasn't for you, I probably wouldn't have entered the world of AGVs. Thanks for that, and for many helpful discussions and advice. When things got rough, I always knew who to talk to. Thank you! *Martin Dahl*; you are one of the nicest people I met at the department. I don't think you are even capable of being mean. Good for you. I also still remember your first presentation in the DK meeting when you had just started at the department. You showed us your house and cats and master thesis work. Believe it or not, I found your style very inspirational. *Anton Albo*; we should have spent more time together. In those sporadic conversations we had, you taught me some really valuable lessons. I did put those lessons into practice, and they just worked. You are a great person and I have enjoyed talking to you very much. Best of luck with your research!

Masoud Bahraini; I wish I had spent more time with you, but that's life. You taught me some good techniques in ping-pong, for which I am grateful! *Constantin Cronrath*; you also taught me many valuable lessons. I needed to discuss a few things with you a couple of times, and you always surprised me with inspirational answers. I really liked your story in a student project where your German supervisor said: *Constantin, I will not have my students in the second row to complain. If something is not working, you take responsibility and fix it. Ramin Jaberzadeh Ansari*, you have always been nice to me. You gave me a very beautiful piece of handicraft, which I still have at home, and you gave me some simulation model of robots, which I have used in my thesis. Thank you for being so nice!

Robert Hult; you were one of the first friends I found in Sweden back in 2011 when I had started my master's. Over the years you have always helped me and gave me guidance when things were rough. Thank you for your support! *Zahra Ramezani*; you have always given me delicious cookies and candies when I was really hungry and in need of calories. How did you know? Thanks for being such a nice colleague and good luck with your research. *Sabino Francesco Roselli*; it was a pure pleasure to meet you. I have had so much fun talking to you and I have learned so much from you. You have been very supportive over the past couple of years. The only thing I never understood is why you always resisted when I wanted to teach you *Scopa*. I thought it would be good to help you become a bit more sociable, but I failed (sigh).

And now, I would like to thank *Oskar Wigström*, my co-supervisor during my maters', and my colleague and coauthor and good friend during my PhD studies. You showed me how a bright and creative person can be unbelievably patient with the ones who don't have all the answers. Thanks for all the good things you taught me, and thanks for the *Dalahäst*. Moreover, I would like to thank *Nina Sundström*

for being a such a nice colleague and friend. I listened to your grandmother's advice on how to deal with difficulties of PhD studies, as she said *just continue*. Apparently it worked! Also, big thanks to *Emma Vidarsson* for kindly teaching me how to run our KUKA robot in the lab although you were very busy with your own thesis. Moreover, special thanks to *Faisal Altaf*, for being so nice and helpful. I really liked your theory of *optimal stress*, about being nervous sufficiently. Finally, big thanks to my former English teacher, *Zhaleh Malek*, who spent a great deal of time helping me improve my writing skills about a decade ago. If I enjoy writing today, it is because of you.

The fantastic administrators at Chalmers *Christine Johansson*; I don't know how many times I messed up the *travels and purchases* form, yet you always helped me fix it. Every time I asked for something you quickly provided it. I am especially grateful for your help in the final months of my work. Thanks for being so kind. Moreover, I would like to thank *Natasha Adler Grønbech* for being the best administrator of the doctoral program. Furthermore, I'm grateful to *Madeleine Persson*, who always kindly helped with the student lists, exam sheets, etc. when I had teachings. Moreover, many thanks to *Marie Iwanow*, our former administrator back in 2011, for her unlimited assistance and support, from the beginning of my studies at Chalmers. Finally, I'd like to thank our former communication officer, *Malin Ulfvarson*. When I was designing a poster for the *Best Poster* contest in the department in 2015, I consulted with you, and you gave me some invaluable advice and design ideas. I also happened to win the award, which was extra fun.

And to my family I want to thank my dear parents and sister, for their everlasting care and support during these years. I don't think after getting my PhD I want to stop teaching myself new things. I just cannot. This is because of the love for learning that you have instilled in me since a very young age. I love you so much.

I also want to thank my beloved wife. You entered my life when I was *almost* finished with my studies, which was about two years ago! Your help was tremendous and highly appreciated. In one instance, I could not decide if I wanted to use C++ or python for my AGV project, and I was just wasting time to make a decision. Then, you did a beautiful cost-benefit analysis for me. This helped me to make my decision and never regret it. I love you!

And to you my dear reader, who have read so far. I hope I have acknowledged you as well. If not, now I thank you for reading my thesis, and for forgiving my forgetfulness!

Sarmad Riazi
Göteborg, March 2020

List of Publications

This thesis is based on the following appended papers:

Paper 1. Sarmad Riazi, Kristofer Bengtsson, Oskar Wigström, Emma Vidarsson, and Bengt Lennartson. *Energy optimization of multi-robot systems.* Proceedings of the 11th IEEE Conference on Automation Science and Engineering (CASE), Gothenburg, 2015, pp. 1345–1350.

Paper 2. Sarmad Riazi, Oskar Wigström, Kristofer Bengtsson, and Bengt Lennartson. *Energy and Peak-power Optimization of Time-bounded Robot Trajectories.* IEEE Transactions on Automation Science and Engineering, 2017, 14(2): 646–657.

Paper 3. Sarmad Riazi, Oskar Wigström, Kristofer Bengtsson, and Bengt Lennartson. *A Column Generation-based Gossip Algorithm for Home Healthcare Routing and Scheduling Problems.* IEEE Transactions on Automation Science and Engineering, 2019, 16(1): 127–137.

Paper 4. Sarmad Riazi, Oskar Wigström, Kristofer Bengtsson, and Bengt Lennartson. *Parallelization of a gossip algorithm for vehicle routing problems.* Proceedings of the 14th IEEE Conference on Automation Science and Engineering (CASE), Munich, 2018, pp. 92–97.

Paper 5. Sarmad Riazi, Thomas Diding, Petter Falkman, Kristofer Bengtsson, and Bengt Lennartson. *Scheduling and Routing of AGVs for Large-scale Flexible Manufacturing Systems.* Proceedings of the 15th IEEE Conference on Automation Science and Engineering (CASE), Vancouver, 2019, pp. 891–896.

Paper 6. Sarmad Riazi, Kristofer Bengtsson, and Bengt Lennartson. *Energy Optimization of Large-scale AGV Systems.* IEEE Transactions on Automation Science and Engineering (2020), *Early access article.*
DOI:1109/TASE.2019.2963285.

Other relevant publications authored or co-authored by Sarmad Riazi:

Sarmad Riazi and Bengt Lennartson. *Using CP/SMT Solvers for Scheduling and Routing of AGVs.* Conditionally accepted as an invited submission of Paper 5 to the IEEE Transactions on Automation Science and Engineering as a Conference-Based Special Issue.

Sabino Francesco Roselli, Fredrik Hagebring, **Sarmad Riazi**, Martin Fabian, and Knut Åkesson. *On the Use of Equivalence Classes for Optimal and Sub-Optimal Bin Packing and Bin Covering.* Invited submission to the IEEE Transactions on Automation Science and Engineering as a Conference-Based Special Issue.

Sabino Francesco Roselli, Fredrik Hagebring, **Sarmad Riazi**, Martin Fabian, and Knut Åkesson. *On the Use of Equivalence Classes for Optimal and Sub-Optimal Bin Covering.* Proceedings of the 15th IEEE Conference on Automation Science and Engineering (CASE), Vancouver, 2019, pp. 1004–1009.

Emile Glorieux, **Sarmad Riazi**, and Bengt Lennartson. *Productivity/energy optimisation of trajectories and coordination for cyclic multi-robot systems.* Robotics and Computer-Integrated Manufacturing, 2018, 49: 152–161.

Sarmad Riazi, Oskar Wigström, Kristofer Bengtsson, and Bengt Lennartson. *Decomposition and distributed algorithms for home healthcare routing and scheduling problem.* Proceedings of the 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, 2017, pp. 1–7.

Oskar Wigström, Nikolce Murgovski, **Sarmad Riazi**, and Bengt Lennartson. *Computationally efficient energy optimization of multiple robots.* Proceedings of the 13th IEEE Conference on Automation Science and Engineering (CASE), Xi'an, 2017, pp. 515–522.

Sarmad Riazi, Kristofer Bengtsson, and Bengt Lennartson. *From trapezoid to polynomial: Next-generation energy-efficient robot trajectories.* Proceedings of the 13th IEEE Conference on Automation Science and Engineering (CASE), Xi'an, 2017, pp. 1191–1195.

Kristofer Bengtsson, Jan-Markus Rödger, **Sarmad Riazi**, Oskar Wigström, Niki Bey, and Bengt Lennartson. *From Tweets to Energy Optimal Robot Stations.* in F. Frank Chen (ed.) *Sustainable production automation.* New York: Momentum Press, 2017, pp. 59–75.

Sarmad Riazi, Kristofer Bengtsson, Rainer Bischoff, Andreas Aurnhammer, Oskar Wigström, and Bengt Lennartson. *Energy and peak-power optimization of existing time-optimal robot trajectories.* Proceedings of the 2016 IEEE International Conference on Automation Science and Engineering (CASE), Fort Worth, TX, 2016, pp. 321–327.

Nina Sundström, Oskar Wigström, **Sarmad Riazi**, and Bengt Lennartson. *"On the conflict between energy, stability and robustness in production schedules.* Proceedings of the 2016 IEEE International Conference on Automation Science and Engineering (CASE), Fort Worth, TX, 2016, pp. 1263–1269.

Bengt Lennartson, Kristofer Bengtsson, Oskar Wigström, and **Sarmad Riazi**. *Modeling and Optimization of Hybrid Systems for the Tweeting Factory.* IEEE Transactions on Automation Science and Engineering, 2016, 13(1): 191–205.

Bengt Lennartson, Oskar Wigström, **Sarmad Riazi**, Kristofer Bengtsson. *Energy and peak-power optimization of existing time-optimal robot trajectories.* Proceedings of the 5th IFAC Conference on Analysis and Design of Hybrid Systems (ADHS'15), Atlanta, 2015, pp. 351–357.

Sarmad Riazi, Payam Chehrazi, Oskar Wigström, Kristofer Bengtsson, and Bengt Lennartson. *A Gossip Algorithm for Home Healthcare Scheduling and Routing Problems.* Proceedings of the 19th World Congress of The International Federation of Automatic Control (IFAC), Cape Town, South Africa, 2014, pp. 10754–10759.

Sarmad Riazi, Carla Seatzu, Oskar Wigström, and Bengt Lennartson. *A Gossip Algorithm for Home Healthcare Scheduling and Routing Problems.* Proceedings of the 18th Conference on Emerging Technologies and Factory Automation (ETFA), Cagliari, 2013, pp. 1–6.

List of Acronyms

AI	–	Artificial Intelligence
CAD	–	Computer Aided Design
CG	–	Column Generation
CP	–	Constraint Programming
CSP	–	Constraint Satisfaction Problem
FMS	–	Flexible Manufacturing System
HHCSRP	–	Home Healthcare Scheduling and Routing Problem
HMVRP	–	Heterogeneous Multi Vehicle Routing Problem
JSP	–	Job-shop Problem
LBBD	–	Logic-based Benders Decomposition
LP	–	Linear Programming
MILP	–	Mixed Integer Linear Programming
NLP	–	Non-linear Programming
OR	–	Operations research
SAT	–	Boolean Satisfiability Problem
SMT	–	Satisfiability Module Theories
TSP	–	Traveling Salesman Problem
VRP	–	Vehicle Routing Problem
VRPTW	–	Vehicle Routing Problem with Time Window

Contents

Part I

Introductory chapters

Chapter 1

Introduction

If you see the red traffic light ahead, you do not keep the allowed speed. Instead, you reduce the speed hoping that you will avoid to stop. Imagine you are driving at 40 km/h and suddenly the yellow light ahead of you turns red. Since you are reasonably away from the junction, you probably decelerate gently, hoping the light turns green when you reach the junction. If all goes well you will avoid a complete stand-still, which helps to preserve the momentum and save some fuel. There is no point in driving fast if you are to be delayed behind the light. It is just bad for the environment. In the same way, there is no point for industrial moving devices to move fast when some of them are going to be blocked by others and wait. The fact is, however, that it is just so simple to program such devices to accelerate quickly to high speeds, and then to decelerate aggressively when they should stop. In many industries that are ever more concerned with their carbon dioxide footprint, the common motto is still *keep it simple, make it work*, which results in heavy reliance on such energy inefficient motion profiles.

Aiming for simple solutions that work manifests itself in flexible manufacturing systems. There, the challenge is assignment and sequencing of activities on resources, which has a combinatorial nature. The difficulty is intensified when, for example, mobile resources are also involved, such as automated guided vehicle (AGVs). This requires conflict-free routing and scheduling. The coordination of AGVs, with good productivity in mind, is often a cumbersome challenge, requiring much attention and effort. This leaves little room for more *luxurious* amenities, such as energy efficiency, which on the other hand motivates us researchers to strive for improvements in existing systems.

This thesis highlights our efforts in energy and route optimization of moving devices. Of particular interest are (1) industrial robots in a multi-robot environment, (2) generic vehicles in a Vehicle Routing Problem (VRP) context, and (3) Automated Guided Vehicles (AGVs) in a large-scale Flexible Manufacturing System (FMS). The motivation of our work is addressing problems that are of *industrial need* and of *academic research challenge*.

1.1 Background

Energy optimization of robots

The author's work on energy optimization of industrial robots started in the winter of 2014, within a project funded by the European Union, called Automation and Robotics for EUropean Sustainable manufacturing (AREUS) [1]. The project's aim was to provide solutions to reduce the energy consumption of automotive companies. This and similar projects indicate the concerns of manufacturing industries as primary energy consumers about the energy cost of operating industrial robots as well as their Carbon dioxide footprint.

Among notable examples of earlier efforts are [2], which employed velocity balancing to reduce energy consumption, and [3] that targeted total energy consumption of robot systems, which was then followed by [4], which employed dynamic programming to compute energy efficient trajectories to use in a high-level scheduling scheme.

To get a better picture of the energy consumption of industrial robots, note the following examples. As reported in [5, 6], in a typical body shop, over 500 robots assemble roughly 300 to 500 parts using a total of 3500 to 5000 spot welds before they are dispatched to the paint shop. Furthermore, an average 200 kg payload body shop robot consumes a yearly 8 MWh [7]. As reported in [8], in production processes, robots consume approximately 8% of the total electrical energy consumption.

When we were working with the project AREUS, we realized that among the body of studies, few papers addressed non-intrusive energy optimization of existing plants, that is, without changing plant's configuration, or affecting production rate of the existing robots. This was one of our motivation to contribute to the field in that sense. As a part of this doctoral thesis, in [9], we provided an effective energy reduction technique through a simple procedure; acquiring the original trajectories from a plant, optimizing them with respect to physical and time constraints; feeding back the optimized trajectory. In that paper the reported energy savings were estimations based on electric currents, internally measured by the robot. This thesis encloses [9] as Paper 1.

Later in [10] we refined our procedures further. We employed accurate energy measurement equipment and provided procedures to have reliable measurements and replicable experiments, along with a number of new modifications to our optimization model to target not only the energy consumption, but also the peak-power. One significance of our work in [10] is that the optimization method has been evaluated in many single and multi-robot scenarios, and on a variety of robots with different sizes. The important characteristics of our optimization technique are simplicity of implementation, preservation of path and cycle time, and the fact that no dynamic model is required in the optimization. This makes our work or its variations more likely to be adopted by the industry. In this thesis, our article [10] is enclosed as Paper 2.

Home healthcare routing and scheduling

Our research on the vehicle routing problem with application to home healthcare industry started since mid 2014, and was based on the author's article on the application of Benders decomposition and the Gossip algorithm to heterogeneous vehicle routing problems [11]. The work was put on hold due to the immediate need to work within the project AREUS, and it was later resumed in the summer of 2017. The drive for the project stemmed from the fact that the demand for home healthcare (HHC) services has been increasing due to various reasons such as population aging and client preferences. In Europe, for example, between 1% and 5% of the total public health budget is spent on HHC [12].

Creating daily work schedules for caregivers is a challenging task that should consider many complicating requirements. In many cases this is still done daily by the staff with minimum usage of support tools. One step towards automation of this task is developing mathematical models and algorithms. In the literature, the home healthcare scheduling and routing problem (HHCSRP) is about finding the best route for each caregiver, while a set of side constraints are satisfied.

It is noteworthy that the HHCSRP was first described in [13]. It can be modeled as an extension of the classical vehicle routing problem with time windows (VRPTW) and some extra constraints. In fact, there are many variations of the problem formulation, as highlighted in a survey article [14], where the authors give a comprehensive overview of the recent models and algorithms developed for the HHCRSP, along with common constraints that model specifications given by HHC organizations, patients, and caregivers.

Many of the developed solution methods for HHCSRPs are based on the techniques for Vehicle Routing Problems (VRPs), which is a huge topic itself. See for, example, a detailed review of formulations and exact algorithms for VRPTWs in [15]. According to a survey [16], the best exact algorithms for VRP and VRPTW were based on column generation (CG) algorithms, for example as in [17]. These algorithms are based on the famous work of Desrochers et al. [18]. Apart from the exact solution methods, there exist successful heuristics and metahuristics to handle different variations of the VRP. In a seminal survey [19], the authors analyzed 64 heuristic algorithms in detail, and identified concepts and main algorithmic design-principles prevalent in successful methods as the "winning strategies". Among such strategies are relaxation, decomposition, matheuristics (the incorporation of mathematical programming techniques in a metaheuristic framework), presence of randomness in search moves, and parallelization, to name a few.

As mentioned earlier, we had some experience with application of decomposition and distributed algorithms for a variation of the VRP [11]. Our goal was to utilize that experience and contribute to the field by synthesizing an algorithm based on column generation and gossip algorithms.

In a preliminary publication [20], we used a general purpose MILP solver to solve the local problems of gossip algorithm, called gossip-CPLEX, in the context of HHCRSP. In a subsequent iteration [21], we showed the potential of turning the gossip algorithm into an effective technique for large HHCRSPs by employing a CG-based local solver.

Later in [22], we extended the work, and provided extensive numerical experiments to show the effectiveness of the column generation-based gossip algorithm. For large problem instances, the algorithm is able to outperform the standard CG. We also showed that the framework gives the flexibility for parallelism. Furthermore, in [23], we utilized this and generated a parallelized version of our Gossip-CG algorithm with promising results. These last two articles are included as Paper 3 and Paper 4.

AGV routing and scheduling

The idea of working on the scheduling and routing of AGVs was conceived as a way to bridge our earlier works on energy optimization and vehicle routing. The work started in the spring of 2018, after initial contact with a Swedish AGV manufacture was made. The company is called AGV Electronics (AGVE). The research was fueled by the increasing need for more AGV systems in flexible manufacturing systems (FMSs), as well as the pressure for more sustainable production.

The problem of conflict-free routing and scheduling of AGVs in large-scale manufacturing systems has been an ever-present challenge for many AGV companies. Although these companies have developed rather efficient control policies and algorithms, retrofitting the existing heuristic to future's more dense, more complicated, and more demanding AGV layouts, is not guaranteed to be easy. Furthermore, the installed system will not necessarily be as efficient as expected. Currently, it is common to use heuristics to allocate vehicles to orders and route them. There are also rules of thumbs to avoid collisions and deadlocks.

However, with an increasing demand for high-performance AGV solutions, it is of interest to employ optimization algorithms that handle the order allocation, scheduling, and routing in a more efficient way. Other than that, very little attention has been paid to making the AGV routing and scheduling more energy efficient, not only in the industry but also in the research community. In this thesis, we aimed to present an improved method to tackle the above-mentioned issues, with promising results. We will now give a brief overview of the situation.

One major challenge in energy optimization of FMSs is the sheer size of the real systems, which renders classical optimization methods ineffective, and leaves little room for less immediate demands such as energy optimization. In fact, the subject has been overshadowed by researcher's heavy focus on developing coordination methods for AGVs mainly targeting productivity. For example, see [24], for a review on methods for dispatching, scheduling and routing of AGVs.

Common optimization methodologies for scheduling and routing of AGVs can be categorized into exact and heuristic mathematical methods, simulation studies, metaheuristic techniques and artificial intelligence (AI) based approaches, as surveyed in [25]. From another point of view, the common optimization strategies for AGV systems can be divided into two groups: centralized methods and decentralized ones. In centralized methods, a single controller tackles task assignment and routing of the vehicles, as in [26], whereas in a decentralized strategy these roles are partly or entirely delegated to the vehicles. For example, the authors in [27] propose a decentralized method, where first the vehicles assign tasks to themselves such that a

global objective function is minimized, namely the total completion time. Then the vehicles move towards their destinations using a decentralized algorithm.

Due to our interest and background in mathematical optimization and constraint programming, in the development of our methodologies we have been mainly concerned with such methods, particularly, a Benders decomposition method for AGVs described in [26].

In [26], the authors employed a variation of Benders decomposition [28] to solve AGV scheduling and routing problems in two stages for rather small problems. Their approach tackled the task allocation and scheduling using a constraint programming (CP) model with tardiness as its cost function, and the routing problem was handled using a Mixed Integer Linear Programming model (MILP), which was basically a Constraint Satisfaction Problem (CSP). Their proposed CSP model, however, has difficulty scaling to large systems. Furthermore, the method is limited to AGV layouts with special design, namely those with equidistant nodes on the tracks.

To contribute to the field, in one publication [29], we introduce several extensions and improvements to [26] to allow solving larger problems. In [29] we test our algorithms on a real large-scale AGV layout with hundreds of nodes and arcs, with promising results.

As mentioned, in the field of AGVs for manufacturing systems, the literature addressing the energy consumption is scarce, although the energy consumption of mobile robots in general has received much larger attention, see for instance [30] and [31]. In one of the few publications addressing the energy optimization of AGVs [32], the authors formulate the problem as a vehicle routing problem (VRP), combined with an energy consumption model of the AGV that takes into account the load. However, results of such analysis cannot be readily used to control the vehicles, mainly due to lack of regards for time and collision between vehicles.

Our other contribution to the field, regarding the energy consumption of AGVs, is in [33], where we show that by implementing the effective scheduling and routing algorithm developed in [29], it is possible to improve system efficiency so much that it allows for reduction of speed to save energy while still outperforming the original solution. An important advantage of such energy reduction method is simplicity of its implementation in the real system, since regulating the vehicle's speed can be done conveniently. The method leads to significant improvements in key performance measures such as makespan, as well as energy optimization. Both of our contributions have been developed using a real FMS, namely a plant at Volvo Cars, and are enclosed in this thesis as Paper 5 and Paper 6.

1.2 Research approach

Depending on the problem at hand, we have adopted slightly different research approaches. First, we start with the commonalities between the methods. A literature survey is the initial step to identify articles with most relevant and promising methodologies. At this step, we also assess how easily the potential methodologies can be extended to capture the nature of our problem.

The next stage is often developing an optimization model of the problem, which may be carried out for example through Mathematical Programming or Constraint Programming, as described in Chapter 2. It is often the case that some features of the problem are not captured in this stage, either purposefully, to start with something simpler, or unintentionally, due to lack of enough knowledge of the system at hand. In the latter case, sometimes this knowledge is obtained in next stages, which is then added to the model, for example, as a constraint.

The next crucial step is developing algorithms to solve the optimization model. It is desirable, but not always attainable, to solve the models directly using general-purpose solver in a *reasonable* amount of time. If this can be achieved, as in the case of the robot energy optimization problem in Chapter 3, we may continue with the next step. Otherwise, we should resort to more specialized approaches, such as decompositional algorithms.

The next step, depending on the nature of the problem, is to design *case studies*. Here, the interpretation of a case study is an analysis performed on a micro level. For example, it could be a special instance of the problem designed to demonstrate the performance of the algorithm or the model itself, or a standard benchmark instance. If the problem involves practical experimentation and measurements, the experiments are designed at this stage.

The difference between our methods emerges at this level. If the problem leans towards the theoretical side, as in the case of routing problems in Chapter 4, extensive numerical experiments are carried out over benchmark problems, to assess the effectiveness of the algorithms. However, if the aim is to assess the performance of the methodology for a particular industrial setup, then a few practical tests are performed, as in the case of the robot energy optimization problem in Chapter 3. Sometimes, tests may involve simulations and use of energy models, if it is challenging to perform them on the actual environment, as in the case of the AGV routing and scheduling problem in Chapter 5.

1.3 Research questions

One way of formulating a research question is by synthesizing an *industrial need* together with a *academic research challenge*. Following this recipe, and given the industrial needs and research challenges expressed earlier, the following research questions are defined.

RQ1: How can robot's energy consumption be reduced in a non-intrusive way, while preserving original path and cycle time?

RQ2: In what ways can energy and peak-power for multi-robot cell be reduced?

RQ3: In the context of routing of moving devices, how can a known efficient algorithm be integrated with a metaheuristic to solve larger problems?

RQ4: How can the performance of the integrated optimization approach developed in response to *RQ3* be improved by better utilization of existing hardware?

RQ5: How can conflict-free routing and scheduling of moving devices be addressed using general-purpose solvers for large-scale problems?

RQ6: How can the energy saving methodology developed in response to *RQ1-RQ2* be reused in the context of large-scale routing and scheduling of moving devices?

1.4 Main contributions

This thesis contributes by providing answers to the research questions posed earlier. In particular, Paper 1 addresses *RQ1*. Furthermore, *RQ2* is covered in Paper 2. In Paper 3 the research question *RQ4* is treated, while Paper 4 deals with *RQ5*. Finally, *RQ6* and *RQ7* are dealt with in Paper 5 and Paper 6, respectively.

More specifically, in Paper 1, an optimization model is proposed that aims to minimize energy consumption of industrial robots with promising results, without requiring secret robot parameters. Later in Paper 2, we extend and consolidate the results of the first paper and provide extensive test results to validate our method. We presented different cost functions to minimize energy consumption and peak-power. We showed that our method reduced up to 30% of energy consumption and up to 60% of peak-power. One significance of this work was that the optimization method was evaluated in a wealth of single and multi-robot scenarios, and on a variety of robots. The important characteristics of our optimization technique are simplicity of implementation, preservation of path and cycle time, and the fact that no dynamic model is required in the optimization.

In Paper 3, we combined a well known algorithm for routing problems with a metaheuristic and showed that it could perform better than the first algorithm alone. The metaheuristic is suitable for parallelization, and that is what we achieved in Paper 4.

In Paper 5, we proposed a new heuristic as well as several improvements to an existing approach based on Benders decomposition for solving the conflict free scheduling and routing of automated guided vehicles (AGVs), with promising results. One key feature of our work was that it used data from a real large-scale FMS.

Later in Paper 6, the mathematical model proposed in Paper 5 is extended with additional side constraints to take into account the requirements of the real FMS that we studied. Moreover, we provided a low-level energy consumption analysis of the AGVs and demonstrated that the cruise velocity of the AGVs is the crucial parameter in energy consumption. We also demonstrated the effectiveness of our optimization method on the mentioned FMS. By using our optimization method, the makespan can be reduced up to 40%, while reducing energy consumption by 14%. Finally, we employed the idea of slowing down the vehicles, and we showed that by using our method, it is possible to reduce the energy consumption by around 38%, while key performance indexes such as makespan, lateness, and tardiness remain better than those obtained from the existing traffic controller developed by the AGV manufacturer.

1.5 Outline

This thesis consists of two parts. Part I is a general introduction to the field and puts the appended papers into context. Part II contains the appended papers. The first chapters, Chapter 1 and Chapter 2, give the background and preliminaries required to comprehend the contributions of the thesis. Particularly, mathematical programming and constraint programming and their mutual relationship are discussed in Chapter 2. Moreover, the employed optimization algorithms are reviewed, most notably the decompositional and heuristic methods. In Chapter 3, 4 and 5 the main three problems addressed in the research and the pertaining algorithms are presented, namely the energy optimization of robots, vehicle routing and scheduling, and energy efficient AGV routing and scheduling. Each of these chapters also includes a description of our proposed methods for tackling each problem. Chapter 6 summarizes the appended papers and links them to the research questions defined earlier. Finally, the thesis is concluded by our final remarks in Chapter 7, including some proposals for future work.

Chapter 2

Preliminaries

In this chapter, we give a basic overview of the tools we used to build the models of the problems tackled in this thesis. In particular, mathematical programming, constraint programming, and satisfiability modulo theories are presented. First, for mathematical programming, the special cases of linear programming (LP), mixed integer linear programming (MILP) and nonlinear programming (NLP) are reviewed. Then, constraint programming (CP), which is another paradigm for modeling and solving optimization problems follows. Finally, satisfiability modulo theorem (SMT), which is a more general framework for modeling and solving satisfiability and optimization problems, will be reviewed.

2.1 Basic Concepts

This section aims to provide the reader with the theoretical background on some of the optimization algorithms we have used in this thesis. We begin by introducing basic concepts in the context of constraint-directed search methods. An example of this is the Benders decomposition method that will be described in Chapter 4. Next, *nogoods*, as a key component of these algorithms are studied. In short, nogood is a constraint that is generated to forbid the exploration of an already explored part of the search space in an optimization problem. Inference duality, as a tool for generating nogood constraints, will also be briefly introduced. However, in this thesis, we limit ourselves to simple nogoods, which do not rely on inference duality. It is noteworthy that most of our exposition in this chapter is based on the book by John N. Hooker [34].

Consider the following minimization problem

$$\begin{aligned} &\min\ f(x) \\ &\mathcal{S} \\ &x \in D \end{aligned} \tag{2.1}$$

where $f(x)$ is a real-valued objective function, $\mathcal{S}$ is the set of constraints, x is a tuple $(x_1, ..., x_n)$, and the domain of x is D. A *solution* to the problem is any $x \in D$, and it is called a *feasible* solution if it satisfies all constraints in $\mathcal{S}$, otherwise it is *infeasible*.

The set of all feasible solutions of the problem is called the *feasible set*. If there is no feasible solution to a problem, the problem itself is called infeasible. An *optimal* solution is denoted by x^* and is the one for which $f(x^*) \leq f(x)$ for all feasible x. If a problem is infeasible, it is the convention to let its optimal value be ∞ (or $-\infty$ for a maximization problem). If none of the feasible values of x can provide a lower bound for $f(x)$, the problem is *unbounded* and the optimal value of $f(x)$ is $-\infty$ (or $+\infty$ for a maximization problem). The problem is pronounced *solved* either when the optimal solution is found, or it is proved to be infeasible, or unbounded. To solve an optimization problem, three processes called *search, inference* and *relaxation* work together, forming the components of the optimization scheme. Next, we will study these components.

Search When the police is looking for a fugitive, to facilitate the operation, they normally divide the region into a few sectors by imposing boundaries and setting up perimeters, and then search each sector for the felon. Likewise, to facilitate solving an optimization problem, it can be divided into a series of problem *restrictions* by adding constraints to the original problem P. Then, *search* is conducted by solving each problem restriction denoted by $P_1, P_2, ..., P_m$. These restrictions are *exhaustive* if the union of their feasible sets is equal to the feasible set of the original problem. Hence, searching an exhaustive set of restrictions leads to an *exhaustive search*. Therefore, to find the global optimal solution, optimal solutions of all restrictions are recorded and the best one will be picked. In contrast to the exhaustive search methods, there are incomplete search methods in which the problem restrictions may not be solved to optimality.

One basic search method is to define problem restrictions by fixing x to the values in D, and enumerate all possible combinations and check them against the constraints in $\mathcal{S}$. Since it can be done only for very simple problems, in practice, more sophisticated methods are employed to define and process the problem restrictions. Branching and constraint-directed search techniques are among those methods, and we will discuss the latter shortly.

Inference Inference is a fundamental concept in optimization theory that has close ties with search. By inference we mean inferring and deducing new constraints by examining the constraints in each problem restriction, in other words, making implicit constraints explicit. The new constraints can help speed up the search by ruling out some parts of the search space yielding no better or infeasible solutions. Search and inference interact by enumerating solutions and deducing new constraints. This mutual relationship can be formalized as *inference duality*. Inference duality is a general concept that has important special cases such as linear programming duality, and Lagrangian duality. However, in this thesis, we do not deal with deriving nogoods using inference duality.

Relaxation When a problem (or problem restriction) is difficult to solve, relaxation techniques can be used. Relaxation works by removing some of the complicating constraints or by replacing the objective function with a lower bound. Dropping the constraints could lead to a larger feasible set, which in turn, may lead to a better optimal solution. Therefore, relaxing a minimization (maximization) problem

always leads to a better or unchanged objective function value that is considered as a lower bound (upper bound) to the main problem (or problem restriction). One common example is continuous relaxation of integer variables in a mixed-integer linear programming (MILP) problem, where discrete variable domains are replaced by continuous ones. Another example could be the *nogoods* in constraint directed search methods. This will be further explained in the end of this chapter, and in Chapter 4, where we will review the Benders decomposition.

2.2 Mathematical programming

In this section we describe the notion of mathematical programming and its different variations, namely Linear Programming (LP), Mixed Integer Linear Programming (MILP), and Nonlinear Programming (NLP). LP is a mathematical method that aims to optimize the outcome of a mathematical model of requirements, expressed as linear relationships. The usage of the term *programming* in this context has been a source of confusion due to its reminiscence of the term computer programming.

In general, mathematical programming is about programming in the sense of planning, hence it does not have to do with computers [35]. Yet, mathematical programming models, or mathematical programs, are frequently coded using computer programming languages and solved by respective algorithms. The solutions give the so-called *plans*. There is indeed a historical reason for the nomenclature.

George Dantzig, the developer of the simplex algorithm for solving LPs, dealt with research problems posed by the United States Defense Department in the post-World War II era. The problems were about planning activities for future conflicts. Hence, the term *program*, which was previously used to describe a plan of activities, became associated with the certain class of mathematical problems in the operations research (OR) literature [36].

Linear programming (LP) An LP problem is the one whose constraints and objective function are linear. An LP minimization problem can be written as

$$\begin{aligned} \min\ z &= \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i,\ i = 1, \ldots, m \\ & x_j \geq 0,\ j = 1, \ldots, n \end{aligned}$$

where c_j is objective function coefficient for x_j, a_{ij} is the constraint coefficient for variable x_j in constraint i, and b_i is the right-hand side coefficient in constraint i. The space defined by the set of constraints is called the *feasible region*. A solution to the problem always lies either inside or on a border of the feasible region.

The same problem can also be written in matrix form as

$$\begin{aligned} \min\ & z = c^T x \\ \text{s.t.}\quad & Ax \geq b \\ & x \geq 0 \end{aligned}$$

where A is the matrix of constraint coefficients for the column vector of variables x, C is the row vector of objective function coefficients, and b is the column vector of right-hand side constraint coefficients.

There are efficient algorithms for LP problems, such as the famous simplex algorithm designed by George Dantzig. The simplex algorithm has an exponential-time worst-case complexity, but in practice, it is usually polynomial-time [37]. Although many LP problems can be efficiently solved by the simplex algorithm, many practical problems result in very large models including many variables and constraints, which become intractable. For some problems with special structure, there exist decomposition algorithms that solve several smaller easier problems, instead of one large problems. An example of such algorithms is Dantzig-Wolfe decomposition for LP problems [38]. An extension of this method is the column generation method [39], which will be discussed in Chapter 4 as it is used in Paper 3.

Nonlinear programming (NLP) A constrained NLP can be formulated as

$$\begin{aligned} \min\ & f(x) \\ \text{s.t.}\ & g_i(x) \geq b_i,\ i = 1, \ldots, m \end{aligned}$$

where $f(x)$ and $g_i(x)$ are nonlinear functions over the vector of variables x.

There are many algorithms for solving constraint NLPs such as penalty function (exterior point) and barrier function (interior point) methods. In penalty function methods, the problem is reformulated as an unconstrained optimization problem. The reformulation is carried out by integrating a penalty term using the constraint to the objective function. The role of the penalty term is to measure and penalize the distance from an infeasible point to the feasible region defined by the constraint set. This penalty term is called the penalty function. Then, starting from a possibly infeasible solution, a series of unconstrained optimization problems are solved and the subsequent infeasible solutions are penalized until the distance from the feasible region becomes very small. That is, the feasible and optimal point is found. Such a search algorithm is also called an exterior method.

Alternatively, instead of penalizing infeasible solutions, one may penalize the deviation from the border of the feasible region. That is, instead of generating a series of infeasible points that lead to a feasible optimal solution, one may generate a series of feasible points (interior points) that lead to the optimal solution. Such a function that makes it more expensive to get close the border of the feasible region is called a barrier function, and the subsequent search algorithm is a called a barrier function or interior point method.

One of the most famous software packages that implements the interior point method is the `ipopt` solver [40]. We used it to solve our NLP model of the energy

optimization problem presented in Paper 1 and Paper 2. An important remark about using this solver is that, supplying the solver with a nontrivial feasible solution can drastically improve its performance and reduce the solution time. As discussed in the above-mentioned papers, to solve our robot energy optimization model, we used the physical solution generated by the robot controller as the initial solution. This enabled us to solve large and difficult NLPs within seconds.

Mixed integer linear programming (MILP) A general integer linear programming (ILP) problem with n variables and m constraints can be formulated as

$$\begin{aligned} \min\ z &= \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i,\ i = 1, \ldots, m \\ & x_j \geq 0,\ \text{integer},\ j = 1, \ldots, n \end{aligned}$$

where the objective function and constraints are linear functions. In this case, all variables are integer-valued. Hence, the problem is a pure integer linear programming (ILP) problem. However, if some of the variables are real-valued, while the others are integer-valued, the problem is called a mixed integer linear programming (MILP) problem.

If the integrality requirement over the integer variables is removed, the model reduces to an LP model. This new model is called an LP relaxation, for which efficient algorithms such as simplex exist. An LP relaxation has a solution that is equally good or better than the solution to the original MILP problem. The solution, however, is not necessarily integral. Thus, LP relaxation needs to be combined with a technique called branching. In branching, variables with fractional values are systematically forced to take either the rounded up or rounded down values that form different branches that will be solved as new LPs.

For example, if a relaxation leads to the solution $x_1^* = 5.2$, then two branches are created. In the first branch, the LP is resolved with a new constraint $x_1 \leq 5$, while the second branch is resolved with $x_1 \geq 6$. Let us denote the best integer solution found at any point as the *incumbent*. The mentioned process is repeated until an integer solution is obtained that is better than the incumbent. It may also be possible to reason that the current branch will not lead to a better integer solution. One way to check that is if the LP relaxation in the current branch has a worse solution than the incumbent solution. In this case, that branch is pruned, i.e. it will not be explored any further. This combination of relaxation and branching is called *branch and bound*.

MILP and ILP models are useful in applications where decisions need to be made. For example, if sequence of operations or assignment of tasks to machines are to be determined, such models can capture the intended behavior. Among notable examples of integer programming problems are traveling salesman problems (TSPs), and vehicle routing problems (VRPs), which come in many forms and extensions. A

VRP can be regarded as an extension of a TSP. Next, we will describe a TSP with an example, and later in Chapter 4, we will explain a VRP model. In this thesis, Paper 3 and Paper 4 deal with a variation of VRP.

Traveling salesman problem (TSP) TSP is one of the most famous combinatorial optimization problems. This problem is easy to express, difficult to solve, and prominent in theory and application. We begin by a general definition and formulation of the problem, and then continue with some notes on its complexity and solution procedures. The problem is stated as follows.

A traveling salesman has to visit n cities. Each city is to be visited exactly once. The salesman has to determine a tour that starts in his home town, visits all cities, and returns to the home town. The aim is to minimize the total tour distance.

A TSP can be modeled as a graph. A graph $G = (\mathcal{V}, \mathcal{E})$ is defined by a set of vertices (nodes) $\mathcal{V} = \{v_1, v_2, v_3, ...\}$, a set of edges (arcs) $\mathcal{E} = \{e_1, e_2, e_3, ...\}$, and a relation between them. Now, consider an example with $\mathcal{V} = \{0, 1, 2, 3, 4, 5\}$, and $\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. In Figure 2.1 a TSP with the starting node 0 and 5 nodes to visit, together with two of the possible tours are depicted. The left figure shows the optimal tour.

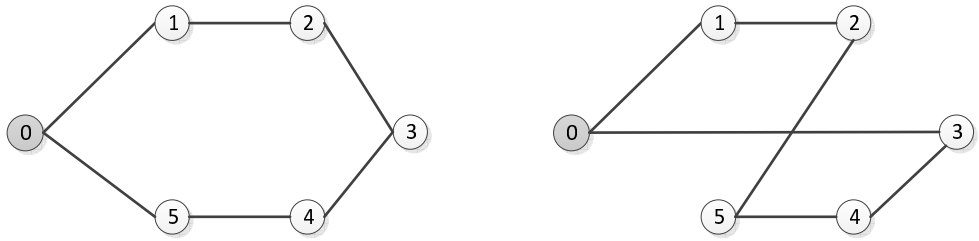

Figure 2.1: An example of a TSP with two possible tours. The left one is the optimal solution.

In an asymmetric TSP, the cost of going from one node to another is different in opposite directions. Therefore, between each pair of nodes, two edges and their corresponding costs are considered. To model an asymmetric TSP, binary variables are introduced as follow:

$$y_{ij} = \begin{cases} 1, & \text{if the edge between the nodes } i \text{ and } j \text{ is selected, } i \in \mathcal{V}, j \in \mathcal{V} \\ 0, & \text{otherwise} \end{cases}$$

Additionally, the cost for edge y_{ij} is denoted by c_{ij}, where c_{ij} does not need to be the same as c_{ji} . For example, in Figure 2.2 (center) the (optimal) solution y^*_{ij} is indicated by $y_{01} = y_{12} = y_{23} = y_{34} = y_{45} = y_{50} = 1$, and the rest of the variables are zero. The optimal solution can alternatively be stated as $y_{05} = y_{54} = y_{43} = y_{32} = y_{21} = y_{10} = 1$, where the tour is taken counter-clockwisely. All possible directed edges, the clockwise tour, and the counter-clockwise tour are shown from left to right in Figure 2.2.

The asymmetric TSP can then be formulated as follows.

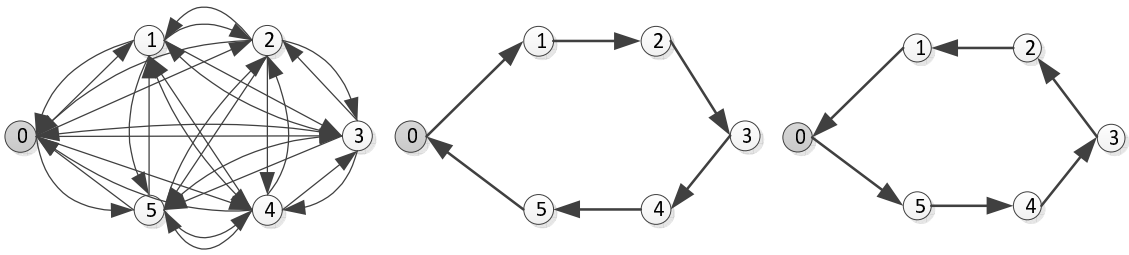

Figure 2.2: From left to right: All possible directed edges, the clockwise, and the counter clockwise TSP tours

$$\min \quad z = \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} c_{ij} y_{ij} \tag{2.2}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{V}} y_{ij} = 1 \qquad \forall i \in \mathcal{V} \tag{2.3}$$

$$\sum_{i \in \mathcal{V}} y_{ij} = 1 \qquad \forall j \in \mathcal{V} \tag{2.4}$$

$$\sum_{i \in \mathcal{S}} \sum_{j \notin \mathcal{S}} y_{ij} \geq 1 \qquad \mathcal{S} \subset \mathcal{V},\ 2 \leq |\mathcal{S}| \leq |\mathcal{V}| - 2 \tag{2.5}$$

$$y_{ij} \in \{0, 1\} \qquad \forall i \in \mathcal{V}, \forall j \in \mathcal{V} \tag{2.6}$$

The objective function (2.2) minimizes the cost of the tour. Constraints (2.3) and (2.4) are *degree constraints*. That is, they ensure that every node is left exactly once, and is entered exactly once respectively. Constraint set (2.5) is called the *sub-tour elimination constraint* (SEC), and its role is to forbid sub-tours, i.e. the tours that only connect a subset of nodes. In other words, SECs eliminate *non-Hamiltonian* tours. Figure 2.3 compares a Hamiltonian cycle satisfying all TSP constraints, with sub-tours that only satisfy constraints (2.3) and (2.4). Finally, constraint (2.6) impose binary conditions on the variables.

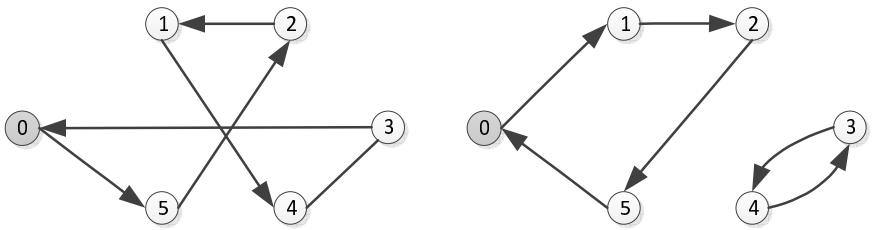

Figure 2.3: Left: a Hamiltonian cycle. Right: a non-Hamiltonian cycle with two sub-tours

Let us examine SECs in more detail. SECs can be formulated in a number of ways, and each formulation has its own interpretation (see more on SECs for TSP in [41]). The formulation we have used here, i.e. (2.5) means that there must be at least one edge going out from the subset $\mathcal{S}$ to the nodes outside. Consider Figure 2.3

(right); we will show how SECs will remove the sub-tours. Suppose that for the node set $\mathcal{V} = \{0, 1, 2, 3, 4, 5\}$ we choose the following subsets: $\mathcal{S} = \{0, 1, 2, 5\}$ and $\bar{\mathcal{S}} = \mathcal{V} \setminus \mathcal{S} = \{3, 4\}$. Then, using (2.5), the SECs particular to $\mathcal{S}$ will be:

$$y_{03} + y_{04} + y_{13} + y_{14} + y_{23} + y_{24} + y_{53} + y_{54} \geq 1$$

This constraint means that there should be at least one edge from $\mathcal{S}$ towards $\bar{\mathcal{S}}$, as shown in Figure 2.4 (left). Suppose that we choose the edge going from node 2 to node 3 (e_{23}). The existence of such an edge means that in node 2 there is more than one outgoing edge, i.e. (2.3) is violated, and in node 3 there is more than one incoming edge, that is (2.4) is violated. To remedy this, e_{25} and e_{43} are removed as in Figure 2.4 (middle), and e_{23} is added. Now (2.3) is violated in node 4 (no outgoing edge), and (2.4) is violated in node 5 (no incoming edge). Hence, adding edge e_{45} will satisfy these constraints and results in a Hamiltonian cycle as in Figure 2.4 (right). One final note about SECs is that sub-tours over one node (and hence over $|\mathcal{V}| - 2$ nodes) cannot happen, due to the degree constraints, that is (2.3) and (2.4). Therefore, SECs are only valid for $2 \leq |\mathcal{S}| \leq |\mathcal{V}| - 2$.

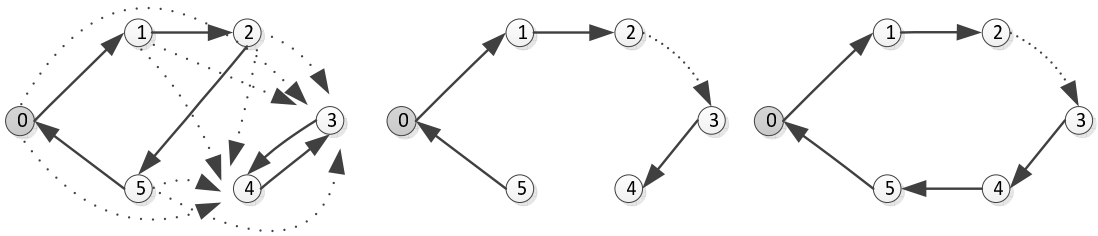

Figure 2.4: Removing the sub-tours using the SEC constraints

For a TSP with n nodes ($|\mathcal{V}| = n$), the introduced ILP model includes $n(n-1)$ binary variables, $2n$ degree constraints, and $2^n - 2n - 2$ SECs. As any other ILP, one natural solution method seems to be a branch and bound algorithm. However, due to the large number of SECs, even for a not so large problem, including all these constraints in a simple branch and bound method quickly becomes intractable. Hence, some specialized algorithms have been developed that first find a lower bound by relaxing SECs, and then methodically include those constraints. In addition to relaxation-based methods that yield exact solutions, for larger problems many different heuristic methods have been developed. Laporte in [42] provides an extensive review of exact and heuristic methods for TSP.

2.3 Constraint programming

Constraint programming (CP) framework is a programming paradigm originating from the computer science and artificial intelligence communities, which aims to solve combinatorial optimization problems. We mentioned in Section 2.2 that the term *programming* in mathematical programming used by the operations research community has a historical reference to activity planning. However, its usage by the CP community does indeed refer to computer programming [36].

CP can be viewed as a two-level architecture that involves two components; constraints, and programming. The constraint component, also known as constraint store, includes variables that enables users to construct constraints over them. The variables represent memory cells in a computer. The second level is about writing a computer program that manipulates the variables such that the constraints are satisfied. Therefore, a constraint program is a computer program designed to solve a problem, and not just a statement of the problem, and constraint programming is therefore a computer programming technique. This is contrast with what a mathematical program and mathematical programming represent [36].

A CP framework, as a programming technique, provides a rich declarative language for stating problems (the constraint component). It is combined with a programming language (programming component) that allows implementation of an algorithm to solve the problem in a procedural way. Hence, CP is an alternative to mathematical programming, and especially to integer programming, for solving combinatorial problems. These problems may either call for finding a feasible solution, or the optimal solution. The former can be regarded as a Constraint Satisfaction Problem (CSP), while the latter is regarded as Constraint Optimization Problem (COP). Next, we will formally define these problems.

Constraint satisfaction problem Consider a finite set of *decision variables* $X = \{x_1, \ldots, x_n\}$, together with respective domains $D = \{D_1, \ldots, D_n\}$, where D_i that is the set of possible values of x_i. In general, there is no restriction on the type of the legal values in D_i, i.e. the variable can be of type integer, real, etc.

A constraint C_i, from the constraint set $C = \{C_1, \ldots, C_j\}$, is a relation R_i defined on a subset of variables $S_i = \{x_{i_1}, \ldots, x_{i_r}\}$, $S_i \subseteq X$. The relation R_i gives the simultaneous legal value assignment to the corresponding variables, which is a subset of the Cartesian product $D_{i_1} \times, \ldots, \times D_{i_r}$.

Instantiation of a variable means that it is assigned a value from its domain. Likewise, an instantiation of S_i is a tuple $\bar{a} = (a_{i_1}, \ldots, a_{i_r})$. An instantiation or tuple $\bar{a}$ *satisfies* a constraint C_i, iff $\bar{a} \in C_i$. That is, iff the instantiation is defined over all the variables in S_i and the components of $\bar{a}$ exist in R_i.

A constraint satisfaction problem (CSP), is therefore defined as a finite set of variables X with respective domains D, together with a finite set of constraints C, each of which is defined on a subset of X. Hence, a CSP can be viewed as a triple (X, D, C). If there exists a tuple $\bar{a}$ that satisfies every constraint in C, it is called a solution to the CSP. In this case the CSP is *consistent*, which is the equivalent term for *feasible* in the operations research community. If no solution exists, the CSP is deemed *inconsistent* (infeasible). The definitions given above are mostly based on [43] and [44].

Solving CSPs Consider a CSP with a set of constraints including $x + y = 3$, where the domain of the variables is the set of natural numbers. One way is to enumerate all possible value assignments using a *backtracking search* scheme, until a solution is found. Backtracking search for CSPs extends a partial instantiation. This has exponential complexity in the best case. One may also observe that $(x, y) = (1, 1)$ is an illegal assignment, the domain of both variables can be *reduced*, leading to a

smaller search space. This did not even require checking other constraints. Moreover, one may *deduce* two additional constraints $x \leq 2, y \leq 2$ from the original one and *propagate* this information, therefore removing another part of the search space. Detecting inconsistent instantiations as early as possible or even during the search saves a lot of time, if it is done efficiently.

Domain reduction can be achieved by *filtering* algorithms that aim to detect illegal value assignments. These algorithms seek to achieve various levels of consistency, for example *arc consistency*, or *path consistency*. Arc consistency means inferring constraints based on a pair of variables, while path consistency extends the arc consistency to constraints including three variables. Arc consistency can be *full*, or *partial*, depending on if all illegal assignment to pairs of variables are removed. There is a trade-off between the effort spent on pre-processing using different consistency levels and that spend on the resulting search tree. The same is true about the level of consistency aimed for during search. It is often practice to aim for partial consistency. Algorithms that achieve a bounded amount of inference are called *local consistency-enforcing*, *bounded consistency inference*, or more commonly, *constraint propagation* algorithms [43]. Efficient propagation algorithms have been developed for certain constraint structures, called *global constraints*, with recurring patterns in different problems.

Global constraints A global constraint states the relationship between a non-fixed number of variables [45, Chapter 7]. Perhaps the most famous global constraint is $\texttt{alldifferent}(x_1, \ldots, x_n)$, which states that the values assigned to the variables must be pairwise distinct, i.e.

$$\texttt{alldifferent}(x_1, \ldots, x_n) = \{(d_1, \ldots, d_n) | \forall i \; d_i \in D(x_i), \forall i \neq j, d_i \neq d_j\}.$$

Although the same relationship can be written using the conjunction of several simpler constraints, due to several reasons it is recommended to use global constraint where possible [46, Chapter 1]. One benefit is that global constraints reduce the burden of modeling and lead to a more readable and compact *program* which is easier to debug. This is because the syntax of a global constraint is shorthand for a frequently recurring pattern. Moreover, global constraints capture certain substructures in the problem for which efficient filtering algorithms exist. Another reason is that, in an integrated optimization scheme where CP and OR methods are used together, global constraints defined over integer variables maybe relaxed to linear programming formulations, leading to more efficiency [46, Chapter 5].

Constraint optimization problem A Constraint Optimization Problem (COP) is a CSP together with an *objective function* that seeks to minimize or maximize a certain performance measure. The objective function can be defined as $g : D_1 \times, \ldots, \times D_n \rightarrow R$, so that at any solution of the CSP, the function $g(x_1, \ldots, x_n)$ can be evaluated. For a minimization (maximization) problem, the solution(s) with the smallest (largest) value of the objective function is deemed optimal. The objective function can be written as the sum of a series of real-valued components, each involving only subsets of the variables, in which case the objective function is referred to as *global objective function*.

CSP/COP: a small example Consider a CSP (X, D, C) where

$$
\begin{aligned}
X &= \{x, y, z, u\}, \\
D &= \{x \in \mathcal{N}, y \in \mathcal{N}, z \in \mathcal{N}, u \in \mathcal{N}\}, \\
C &= \{x^3 + y^3 + z^3 + u^3 = 100, x < u, x + y = z\}
\end{aligned}
$$

with $\mathcal{N}$ being the set of natural numbers. The instantiation $\bar{a} = (1, 2, 3, 4)$ satisfies all the constraints and is therefore a solution to the CSP [44]. The solution $\bar{a}$ is also the optimal solution to the COP defined over the same CSP with the objective function $g(x) = x$ that is to be minimized.

Solving COPs There are several approaches to solve constraint optimization problems. One can always find a solution to a COP by solving a series of CSPs. Such a reduction allows for using all constraint processing techniques available to CSPs. A simple alternative approach is to extend the backtracking search algorithm such that it does not halt after finding the first solution, but continues discovering all solutions. This method can be accelerated by utilizing the objective function to prune parts of the search tree. For instance, given a partial solution to S_i of a minimization problem, the component of the cost function involving S_i is evaluated. If the value is already higher than the incumbent solution, the partial solution should be discarded and the node is pruned because the global objective function will have a worse solution. This scheme resembles the depth-first branch and bound search that is popular in the Operations Research (OR) community, albeit the CP community has developed its own *bounding evaluation functions* to accelerate the search [43, Chapter 13].

2.4 An overview of SAT/SMT

Since we have used the SMT solver Z3 [47], a brief overview of the SAT/SMT technology is given. There are many similarities between satisfiability and CP, and we attempt to draw parallels between them. Much of our disposition in this subsection is based on [48].

In the field of formal verification, the goal is to prove the correctness of a piece of software with respect to a given formal specification. The concept of correctness or *satisfiability* is parallel to the *consistency* in CP and *feasibility* is mathematical programming. In the context of CP, we wish to find a consistent value assignment to the variables of a CSP, or even the best possible assignment to the variables of a COP. In the field of satisfiability we wish to *prove* the *correctness* of a *theory*. A theory is an (in)finite set of *formulas*, which are characterized by common grammatical rules, the functions and predicates that are allowed, and a domain of values. Thus, formulas can be thought of as a parallel to constraints in CP. For example, propositional logic is a theory, and linear arithmetic based on integers is another theory. Every theory is defined over a set of symbols. For instance, linear arithmetic is defined over symbols such as "+" and "≥".

To illustrate better, a formula (constraint)

$$(2x_1 + 3x_2 \leq 4) \vee (x_1 - 2x_2 \geq 5)$$

with continues domain variables is an example of a linear arithmetic theory, while the formula

$$x_1 \wedge (x_2 \vee \neg x_3)$$

with Boolean variables is an example of a propositional logic theory. Note that problems with linear arithmetic constraints and with logical constraints can be solved not only by an SMT solver, but also by traditional OR and CP method.

We mentioned that in CP, there are three main ways of solving a problem, either through propagation, or search, which are sometimes combined together in more sophisticated solvers. Similarly for SAT problems, there are two approaches for formal reasoning; proof-theoretic, and model theoretic. The former is about using an inference system, similar to inference in CP, which deduces new constraints and more information using the existing ones. The latter is about enumeration of possible solutions from a finite number of candidates, which is similar to search in CP. As in CP, these two approaches can be combined. Other concepts such as *assignment* and contradiction in SAT problems are parallel with instantiation and inconsistency. In SAT problems, to check if a formula ϕ is satisfiable, one can also check if $\neg\phi$ is a contradiction.

Now we define the Conjunctive Normal Form (CNF). The basic building block of a formula is called its atoms. For example, in propositional logic, Boolean variables are the atoms. Moreover, a literal is either an atom or its negation. A formula is expressed in CNF if it is a conjunction of disjunctions of its literals. The importance of CNF is that every formula with a Boolean structure can be transformed into a CNF, which is useful for automated theorem proving.

The term Satisfiability Modulo Theories (SMT) refers to satisfiability problems for an arbitrary theory or a combination of such theories. As mentioned earlier, an SMT solver can solve both constraint satisfaction problems and linear programming problems, which makes it a more general framework than either of the two with a richer constraint language. SMT solvers that are based on SAT offer a richer language than just propositional logic. It should be noted that, there is no clear boundary between the fields of logic, mathematical programming and constraint programming. This is especially true because the communities are inspired by each other's algorithms and concepts and implement them. However, there are still clear distinction between the communities. For example, SMT solvers that are based on SAT solvers can solve linear programming problems, yet due to the underlying technology they benefit from formulas with Boolean structure. Furthermore, the focus of mathematical programming is more on optimization than satisfiability.

2.5 Constraint-directed search methods

As mentioned in the beginning of this chapter, search is conducted through solving a series of problem restrictions. But the question is, in what order should the

restrictions be solved? Constraint-directed search methods answer this question by picking the new restrictions based on the knowledge acquired from solving the previous ones, or in other words, by learning from their mistakes. This knowledge is recorded in form of *nogood* constraints that are added to the nogood set, and will be used to choose the next restrictions.

Search When a problem restriction is solved, it either has a solution or it is infeasible. One would like to avoid the same solution, and possibly, similar solutions with no better objective function value. To do this, one or a set of constraints called *nogoods* are generated after solving that particular restriction that will prevent from the same and perhaps similar solutions. When the next restriction is to be chosen, the nogoods generated so far must be satisfied.

In practice, the problem restrictions are expressed in terms of $x = (x_1, ..., x_n)$, and each specific restriction is determined by fixing x to values from D, where $D = D_{x_1} \times \ldots \times D_{x_n}$. The optimal value of the restriction $P(x)$ is denoted by $v(x)$, while the overall optimal value of the problem P is v. Suppose that in iteration k, by solving the nogood set $\mathcal{N}_k$ we get the solution $x = x^k$, and the resulting problem restriction $P(x^k)$ has the optimal value of $v(x^k)$. Now, a new nogood N is generated and added to the nogood set to form $\mathcal{N}_{k+1}$. The new nogood set is then solved, a new solution $x = x^{k+1}$ is found, and the next problem restriction to solve will be $P(x^{k+1})$. Notice that in the iteration zero the nogood set $\mathcal{N}_0 = \emptyset$. Also note that, provided that there is a finite number of restrictions, the constraint directed search is exhaustive.

Two last concepts to introduce are the notions of *selection function* and *processing* the nogoods. A selection function is a criterion by which a feasible solution x is selected among all the solutions satisfying the nogood set $\mathcal{N}$. In some constraint-directed search methods, it is important to process the constraints in $\mathcal{N}$ to facilitate solving the nogood set. A generic constraint-directed search algorithm is given in Algorithm 1.

Let $v_{UB} = \infty$ and $\mathcal{N} = \emptyset$.
Associate a restriction $P(x)$ of P with each $x \in D$, and let $v(x)$ be the optimal value of $P(x)$.
while $\mathcal{N}$ *is feasible* **do**
  Select a feasible solution $x = s(\mathcal{N})$ of $\mathcal{N}$.
  Compute the optimal value $v(x)$ of $P(x)$ and let $v_{UB} = min\{v(x), v_{UB}\}$.
  Define a nogood N that excludes x and possibly other solutions $x^{'}$ with $v(x^{'}) \geq v(x)$.
  Add N to $\mathcal{N}$ and process $\mathcal{N}$.
end
The optimal value of P is v_{UB}.

Algorithm 1: Generic constraint-directed search algorithm for solving a minimization problem P with variable domain D, where s is the selection function, $\mathcal{N}$ contains the nogoods generated so far [34].

Relaxation As discussed earlier, the search in a constraint-directed method is carried out through enumerating a series of problem restrictions. To define the problem restrictions, a solution $x \in D$ that is feasible in the nogood set $\mathcal{N}$ is chosen, and then variables in $x = (x_1, ..., x_n)$ are fixed to the solution values to form $P(x)$. Then by using its optimal value $v(x)$ new nogoods are generated and added to the nogood set for next iterations. So far, for constraint-directed scheme, we have identified the search component. Moreover, processing the nogoods can be thought of as inference, which will be discussed in the end of this section. But what about the relaxation? In fact the nogood set $\mathcal{N}$ can be thought of as a relaxation R.

As mentioned earlier, when a problem is relaxed, a lower bound on the optimal value of its objective function is obtained. To prove that the nogoods can also be interpreted as relaxations, we need to show how they provide lower bounds on the optimal value of the objective function.

As stated earlier in this section, in step k of the search, the nogood set $\mathcal{N}_k$ is solved and solution x^k is obtained. Next, a problem restriction $P(x^k)$ is solved with optimal value of $v(x^k)$. Now a nogood N_{k+1} is constructed to exclude solutions worse than $v(x^k)$ in iteration $k+1$. Suppose that these unwanted solutions are gathered in a set T. So, for any $x \in T$, the nogood should be able to remove them if it is formulated as a *nogood bound* on the optimal value, as it follows [34]:

$$\begin{gathered} v \geq B_{k+1}(x) \\ B_{k+1}(x) = \begin{cases} v(x^k), & \text{if } x \in T \\ -\infty, & \text{otherwise} \end{cases} \end{gathered} \tag{2.7}$$

The nogood bound states that the undesirable solutions ($x \in T$) must result in a restriction $P(x)$ with optimal value at least $v(x)$. In other words, to get better solutions than $v(x)$, solutions in T must be avoided, and hence $-\infty$ is selected for $x \notin T$. An acceptable solution (v, x) happens when it is feasible in the following problem (and therefore satisfies the nogood bounds):

$$\begin{aligned} &\min\ v \\ &v \geq B_i(x),\ i = 1, ..., k \\ &x \in D \end{aligned} \tag{2.8}$$

Therefore, we can see that the optimal value of the problem (2.1) is lower bounded using the nogoods, and that in effect, (2.8) can be thought of as a relaxation R_k to (2.1).

It is important to point out that there is no need to solve R_k to optimality to get the best feasible x^k to construct a problem restriction. In fact, it is enough that we find a $(v^k,\ x^k)$ such that:

$$v_k < \min\ \{v(x^1), ..., v(x^{k-1})\}$$

Although solving R_k to optimality can be useful, but the fact that it is not a necessity allows us to implement some constraint-directed methods even more efficiently.

2.6 Inference

As pointed out earlier, inference seeks to deduce new constraints from the existing ones to speed up the search.The general idea behind the duality between search and inference is that an optimization problem can be conceived as an inference problem; while a minimization problem is meant to minimize the value of the objective function subject to a constraint set, an inference problem aims to maximize the lower bound on the objective function, where the bound is inferred from the constraints set. Hence, in an optimization problem, the search is conducted among the feasible solutions to determine the minimum value, but in an inference problem, inferring the largest lower bound on the objective function from the constraints involves generating some mathematical proof. To summarize, in an optimization problem we find assignments to variables, while in an inference problem we generate proofs.

If the constraint language is rich enough, for example in LP, it is possible to formulate the *inference dual* of the original problem. The dual of an LP in mathematical programming is a special case of inference duality. In some constraint directed search methods, it is possible to formulate nogoods using the inference duality, for instance in Benders decomposition. In this thesis, however, we use simple nogoods that do not rely on inference duality.

2.7 Summary

In this chapter, we provided an overview of some of the tools we have used in this thesis, namely, mathematical programming, constraint programming, and SMT technology. In mathematical programming, we addressed LP, MILP, and NLP, as they have been used in the thesis. The notion of constraint programming was also discussed, together with its differences and similarities to mathematical programming. The concept of global constraints were also discussed. Finally, the important notions of constraint directed search and inference were reviewed.

Chapter 3

Energy optimization of multi-robot systems

This chapter gives an overview of our work in Paper 1 and Paper 2, which deal with energy optimization of industrial robots. While the research in this area had been ongoing in the Automation research group of Chalmers University of Technology, as in [2], [3], and [4], the author's involvement in the topic began with in the winter of 2014, within a project funded by the European Union, called Automation and Robotics for EUropean Sustainable manufacturing (AREUS) [1], as mentioned in Section 1.1.

The project's aim was to provide solutions to reduce the energy consumption of automotive companies. This and similar projects indicate the concerns of manufacturing industries as primary energy consumers about the energy cost of operating industrial robots as well as their Carbon dioxide footprint. A few examples of energy consumption of industrial robots come next. As reported in [5, 6], in a typical body shop, over 500 robots assemble roughly 300 to 500 parts using a total of 3500 to 5000 spot welds before they are dispatched to the paint shop. Furthermore, an average 200 kg payload body shop robot consumes a yearly 8 MWh [7]. As reported in [8], in production processes, robots consume approximately 8% of the total electrical energy consumption.

It is of utility to categorize the existing methodologies that deal with energy efficiency of production systems. In a survey [49], Paryanto et al. classified the recent studies in energy efficiency of production systems into production planning, commissioning process, and process optimization. Production planning is about coordinating and scheduling the robot operations, as in [4, 50]. It also deals with using energy-efficient and operation specific solutions [51, 52]. Paryanto et al. also give examples of possible improvements during the commissioning process, such as reducing idle time and eliminating waiting times [3, 53]. Furthermore, process optimization involves modifying the trajectories, as in [54], to reduce energy consumption.

As mentioned in Section 1.1, during the project AREUS, we realized that few papers had addressed non-intrusive energy optimization of existing plants, that is, without changing plant's configuration, or affecting production rate of the existing

robots. This was one of our motivation to contribute to the field in that sense. As a part of this doctoral thesis, in [9], we provided an effective energy reduction technique through a simple procedure; acquiring the original trajectories from a plant, optimizing them with respect to physical and time constraints; feeding back the optimized trajectory. In that paper the reported energy savings were estimations based on electric currents, internally measured by the robot. This thesis encloses [9] as Paper 1, and aims to answer to *RQ1*.

Furthermore, in [10] we refined our procedures further. We employed accurate energy measurement equipment and provided procedures to have reliable measurements and replicable experiments, along with a number of new modifications to our optimization model to target not only the energy consumption, but also the peak-power. In this thesis, our article [10] is enclosed as Paper 2, and aims to answer to *RQ2*.

We should emphasize that in our two articles we have treated the subject extensively, with multiple case studies and detailed procedures for data collection, optimization, and measurement. The purpose of this chapter is to provide supplementary information to Paper 1 and 2. Hence, we avoid repeating information as much as possible and refer the reader to the articles for full detail.

The organization of the rest of this chapter is as follows. Section 3.1 clarifies and delimits the challenges in energy optimization of robots that we deal with. Section 3.2 discusses common trajectory generation approaches.

3.1 The challenge

Today's robot trajectories commonly involve rapid acceleration, constant velocity, and rapid deceleration, which resembles the famous trapezoidal velocity profile, which will be explained in this chapter. This is often the case since such a profile results in fast execution of operations, and it is easy to program. However, this way of operating the robots is not energy efficient, in-part due to rapid deceleration that wastes the kinetic energy.

There are other reasons for energy-inefficiency of today's robot trajectories. Consider two robots working in the same common work-space, where one needs to wait for the other before accessing the zone. It is often the case that the robot that must wait, quickly reaches standstill from high velocity. This is can be compared to a driver that drives at high speed and then suddenly stops before the red light. Clearly a lot of kinetic energy is wasted in this way. A more energy-efficient alternative is to slow down the robots that must wait for other robots.

Another challenge, which mostly concerns us researchers, is the fact that if we come up with improved robot paths or trajectories, we will have difficulty convincing the robot users to adopt them. Apart from the obvious technical challenges that arise when integrating new methodologies into existing systems, it is also a burden to make sure the new trajectories or paths are collision-free and then to convince the user of this property. Sometimes, the existing robot trajectories involve processes that must not be modified, such as spot-welding, which is a further complication.

Finally, it is important to convince the potential users that the new methodology does not sacrifice productivity of the plant.

Therefore, for us researches to develop methodologies that are appealing to the industry, it is highly desirable to (1) use the existing functionalities of the robots without needing custom software, (2) make sure the cycle time is not violated, (3) preserve the paths and designated processes, through *careful* trajectory generation. Next, we will discuss common ways of trajectory generation, as well as our own proposed technique.

3.2 Trajectory generation

Consider an environment in which a number of robots interact and perhaps have access to a number of shared zones, as in [55]. Assume that each robot has a given fixed path $f : [0, 1] \to \mathbb{R}^m$, where m is the dimensionality of the robot's joint space. Each robot position $\theta(t)$ along its path f is defined as

$$\theta(t) = f(s(t)), \tag{3.1}$$

where $s : [0, T] \to [0, 1]$ is a scalar and normalized path position, such that $s(0) = 0$, and $s(T) = 1$, i.e. the trajectory starts at $t = 0$ and ends at $t = T$.

The trajectory of each robot is determined by its path as a function of time $\theta(t)$ and its higher order derivatives angular velocity $\omega = \dot{\theta}$, acceleration $\alpha = \dot{\omega}$ and jerk $\zeta = \dot{\alpha}$ along the path. The simplest kind of motion is moving from one point to another, such that at both points velocity is zero. In what follows next we discuss a number of approaches to generate a trajectory that realizes that motion.

Trapezoidal velocity profile A trapezoidal velocity profile consists of three segments; constant acceleration with a constant slope, cruise or coasting at constant velocity, and deceleration with a constant slope, as seen in Figure 3.1.

Now, let us define a straight line in joint space from a start configuration θ_{start} to an end configuration θ_{end} as

$$\theta(s) = \theta_{start} + s(\theta_{end} - \theta_{start}), \ s \in [0, 1]. \tag{3.2}$$

Assume that maximum allowable velocity and acceleration on a joint, i.e. ω_{max} and α_{max} are given. A Trapezoidal velocity profile that maximizes the velocity in the constant cruise velocity $v = \dot{s}$ and constant acceleration/deceleration $a = \ddot{s}$ generates the time-optimal straight line motion, provided that it satisfies the following conditions [56]

$$|(\theta_{end} - \theta_{start})v| \leq \omega_{max} \tag{3.3}$$

$$|(\theta_{end} - \theta_{start})a| \leq \alpha_{max} \tag{3.4}$$

However, the trapezoidal velocity profile has the downside that at θ_{start} and θ_{end} and at transition between phases there are jumps in acceleration, i.e. infinite jerk, as seen in Figure 3.1. For practical purposes, a modified version of this profile called

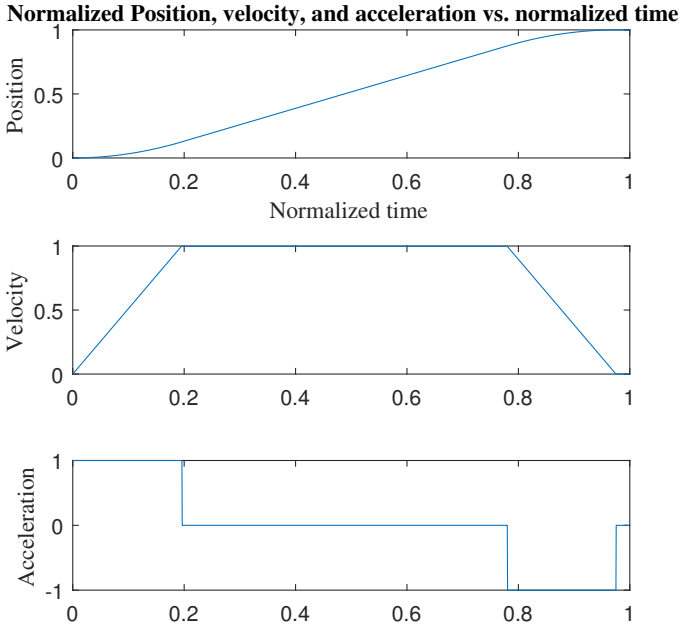

Figure 3.1: Time history of a motion with trapezoidal velocity profile; normalized position, velocity, and acceleration vs. time.

S-curve is used that has constant jerk values at those critical points [56]. Although a trapezoidal profile can be described by a series of first and second order polynomials, in this thesis we reserve the term polynomial for those of higher degrees. A final note on trapezoidal velocity profile is that, although it has the time optimality property under the above-mentioned conditions, it does not take energy consumption into account. However, it is still widely used due to its simplicity.

Polynomial velocity profile As mentioned earlier, trapezoidal velocity profile does not address energy efficiency, and is not smooth at all points. Hence an alternative profile that rectifies those issues is of interest. Let us denote torque by τ. A common performance criterion to minimize heat loss in motor is given in [57], which takes the following simple form for a rigid body rotating around an axis

$$\int_0^T \tau^2 dt \tag{3.5}$$

Assume that the goal is to move a single joint with moment of inertia J from θ_{start} to θ_{end}, such that the above criterion is minimized, and that at θ_{start} and θ_{end}, velocity and acceleration are zero. Then the solution to the differential equation $\tau = J\omega$ has to be a polynomial of degree of at least five [58], as follows

$$q(t) = a_5 t_5 + a_4 t_4 + a_3 t_3 + a_2 t_2 + a_1 t_1 + a_0, \tag{3.6}$$

where $a_0, \ldots, a_5$ are the constant coefficients that are determined by solving the optimization problem described above. Although this motion profile is smooth and energy efficient, in its current form it does not take into account maximum allowable joint velocity and acceleration. Note that in terms of a multi-axis robotic manipulator, the dynamics of the robot can be described as follows [58]

$$\tau = M(\theta)\alpha + C(\theta,\omega)\omega + F_s(\theta)\text{sgn}(\omega) + G(\theta), \tag{3.7}$$

where $M(\theta)$ is the mass matrix, and $C(\theta,\omega)$ is a matrix of Coriolis and centrifugal effects. Matrix $F_s(\theta)$ describes frictional torques, and $G(\theta)$ addresses other effects such as gravity.

Some studies such as [59] incorporate (3.7) in energy optimization, which requires having access to the robot's dynamic model and other classified information that is often restricted. Therefore, the use of simulation tools and system identification techniques become viable, as in [60] and [61]. As an alternative, in Paper 1 we propose a non-linear optimization model for multi-robot systems such that the need to identify robot's dynamic model is eliminated. The technique generates smooth energy efficient trajectories, which will be discussed later in this chapter. Next, we give an overview of our trajectory generation methodology published in Paper 1 and Paper 2.

Trapenomial velocity profile In Paper 1 , we propose an energy optimization procedure that generates smooth energy-efficient trajectories. The trajectories are obtained by feeding trapezoidal (or any other type) trajectories as initial solutions into an optimization problem. The non-linear optimization problem seeks to minimize a performance criterion to reduce energy consumption, and generates trajectories that look like a hybrid of low-order polynomial trapezoidal and high-order polynomial velocity profiles. Therefore, we refer to such a trajectory as *trapenomial* hereafter.

The optimization model does not incorporate a description of robot's dynamics (3.7). However, it directly constrains angular velocity ω, acceleration α, and jerk ζ so that the robot's envelope of operation is not violated. The values depend on the configuration of the joints at each time instance, which is referred to as *pose* or sample of the path.

Consider a set of r robots with paths θ^k, $k = 1, 2, \ldots, r$. We assume that the paths are known at N individual time instances $t_0^k, \ldots, t_N^k$. The poses, velocities, accelerations, and jerks for each robot and individual time instance t_i^k are indicated by θ_i^k, ω_i^k, α_i^k, and ζ_i^k, $i = 0, \ldots, N$.

In Paper 2, we introduce another cost function called *pseudo power*. In Solid Mechanics, the relationship between mechanical power P, angular velocity ω, and torque τ is expressed as $P = \tau\omega$ [62]. In the absence of information about robot's mechanical properties necessary to formulate torque, instead we use angular acceleration, which is roughly proportional to torque. Hence, the multiplication of angular acceleration and velocity at each time instance is regarded as pseudo power, i.e. $\psi_i^k := \alpha_i^k \circ \omega_i^k$, where $\circ$ = Hadamard (entry-wise) vector multiplication. The cost function can now be expressed as

$$\sum_{k=1}^{r}\sum_{i=0}^{N-1}(w_{\psi,i}^k \circ \psi_i^k)^{\mathrm{T}}\psi_i^k\Delta_i^k, \tag{3.8}$$

where $w_{\psi,i}^k$ is the corresponding weighting vector for the pseudo power ψ_i^k. For the full optimization model see Paper 1, and for more information about performance comparison of different objective functions see Paper 2.

3.3 Understanding the effect of optimization

In this section we explain how our optimization procedure works. In particular, we explain how it modifies the original trajectory, and how the result compares with the original one. In the articles, we have already provided real extensive case studies with somewhat complicated trajectories involving all joints. In this section, however, we use simulation of a simple robot motion to understand how the optimization works.

The procedure for minimization of energy consumption of KUKA robots is illustrated in Figure 3.2. It starts with running the programmed motions, and at the same time the trajectory is logged via existing functions in KUKA Robot Language (KRL). As mentioned in Section 3.1, it is desirable to use the existing robots' functionalities to design an optimization procedure. The recorded trajectory will then be used as initial solution to the optimization model described earlier. The optimization is done off-line, and the results are then post-processed to be able to run again on the robot. Note that we use the nonlinear solver ipopt [40], for which we have provided some information in both articles, and in Section 2.2.

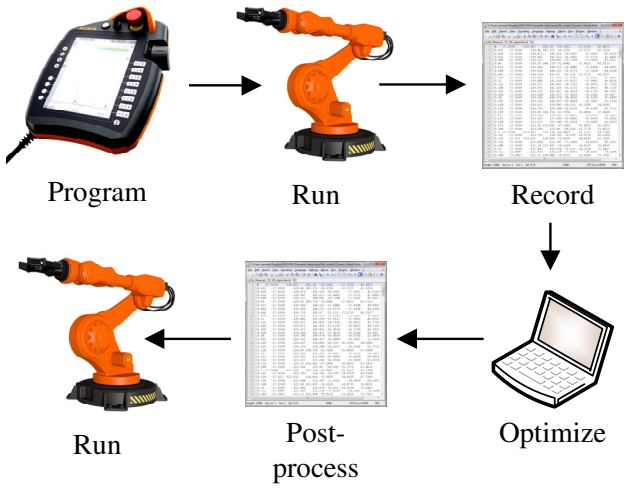

Figure 3.2: Optimization procedure for an existing trajectory/robot

Trajectory optimization with preserved path To better explain the concept of optimizing the trajectory while preserving the path, the following analogy helps. Imagine a car being driven on the road. The car represents the robot, while the road represents the path. Our aim is not to change the road, but the way the car is being driven on it, i.e. the trajectory. Now, consider Figure 3.3 (top). The car starts its motion at time $t = 0$. In the first two seconds, the speed is relatively low, but then the car picks up speed, which can be seen as the larger traveled distance between $t = 3$ and $t = 2$, compared to $t = 2$ and $t = 1$. At $t = 4$ the car breaks and then comes to a standstill at $t = 5$. Note that, in this scenario, the position is recorded at a sampling rate of $\delta t = 1$ and the travel time is 5 seconds.

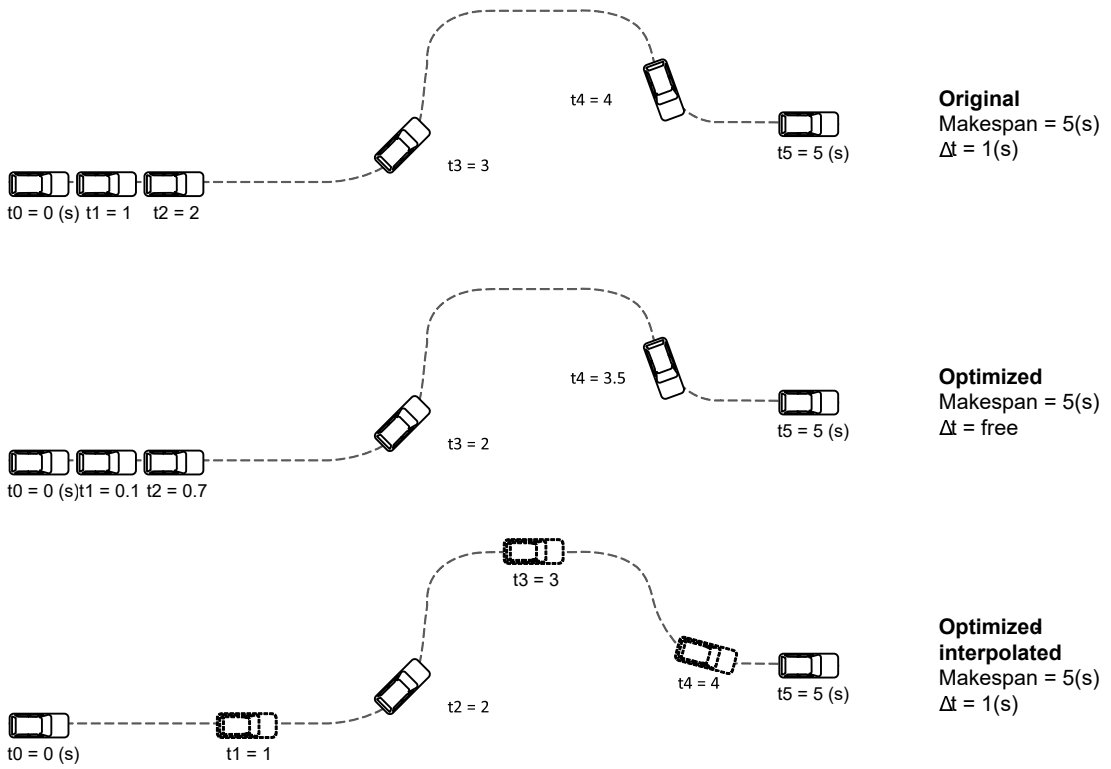

Figure 3.3: The analogy between robot's path and trajectory, with a car, road, and driving profile.

Next, the recorded positions position are fed into the optimization model proposed in Paper 1. With a cost function that minimizes acceleration and a constraint that restricts the travel time to 5 s, the solution is obtained which is depicted in Figure 3.3 (middle). Note that in the solution, the vehicle is still traveling on the same road, and it visits exactly the positions recorded in from the original motion. However, the times at which these positions are recorded have been modified by the optimizer, and the sampling time is no longer $\delta t = 1$ s.

To be more specific, one may note that, in the optimized motion, the vehicle accelerates faster in the slow phase, i.e. at beginning of the motion. This is seen as the same traveled distance during $t_2 = 0.7$ s, compared to two seconds in original motion during . This saves some time for the vehicle, which lets it visit the same location at the first curve, at $t_3 = 2$, i.e. one second earlier compared to the original motion, although the vehicle has 1.3 s to travel between samples at t_2 and t_3, which means it can accelerate more gently compared to the original solution. Now, the vehicle has also more time to decelerate more gently in the final phase of the motion. This is in contrast to the original motion, where the vehicle wasted time in the early phase of the motion, and then accelerated quickly, and after the cruise phase decelerated quickly. In other words, the optimization of acceleration leads to a more balanced motion profile.

As mentioned earlier in Section 3.3, a post-processing phase must be undertaken to change the sampling time back to original, which is done via interpolation. The post-processed solution is shown in Figure 3.3 (bottom). Note that, the travel time is

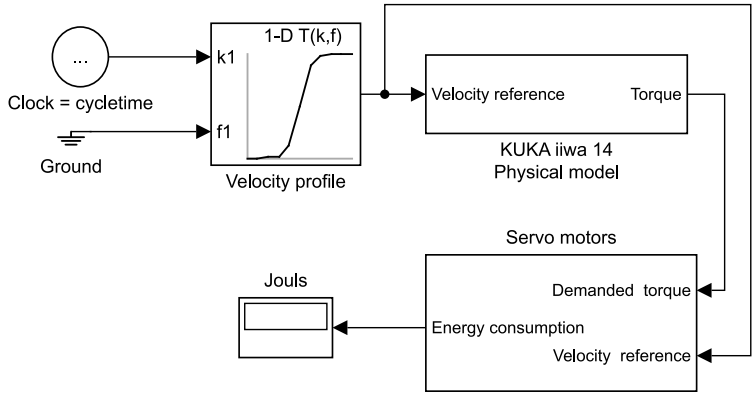

Figure 3.4: The schematic of the robot model in Simulink.

unchanged, and that the car is still on the same road, however, due to optimization of acceleration, the velocity is now better balanced during the motion.

Next, we begin by explaining the simulation model used to compare performance of different trajectories. It is then followed by a description of the baseline trajectory, and then the simulation results will be discussed.

Simulation model To conduct experiments one needs to use either real robots or resort to their models. In case of using a model, one might identify robot parameters using methods given in for example [63, 64]. We have developed a simulation model in MATLAB/Simulink based on the procedure described in [65]. The modeled robot was a KUKA LBR *iiwa 14 R820*.

To prepare our simulation model, first, CAD drawings of the robot's parts were obtained from *kuka_experimental* package developed within *ROS Industrial* project [66]. The CAD drawings were then imported to Simscape Moltibody environment [67] for assembly. The robot's links were then assembled and suitable mechanical properties such as moments of inertia were assigned to each joint. The robot's servo motors were modeled as permanent magnet synchronous motors (PMSM), as described in [65]. The simulation model accepts reference velocity profiles as input to the model of the robot structure, which in turn generates torque profiles. The demanded torques and reference velocities are then fed to their respective servo motor's model. The servo motor's model yields information about stator currents that are used to calculate energy consumption. The energy consumption of any servo motor is calculated by integrating its total input power along its trajectory. This can be achieved using the following formula [65]

$$E = \int_0^T [\frac{3}{2}Ri^2 + \omega_m(\tau + B\omega_m + J\frac{d}{dt}\omega_m)]dt \tag{3.9}$$

where T is cycle time, τ and ω_m are in turn mechanical torque and rotor speed, B is the coefficient of friction, J is the rotor's inertia. Moreover, the term $\frac{3}{2}Ri^2$ models the copper losses. A schematic of the simulation model is depicted in Figure 3.4.

Test trajectories Our goal is to get a clear understanding of how different velocity profiles affect energy consumption. Therefore, we opted for a simple one-joint sweeping motion as the baseline trajectory, as seen in Figure 3.5. The baseline trajectory follows a trapezoidal velocity profile, such that each of acceleration and deceleration phases take up 20% of the 2-second motion time, and the rest of the time is spent on cruising at constant maximum velocity specified by the robot manufacturer. An extra stagnation duration of 0.06 s with zero velocity is padded at the end of the deceleration phase to make sure the higher derivatives of velocity, i.e. acceleration and jerk, also come to zero. This is necessary to ensure that torque on the joint gracefully reaches zero.

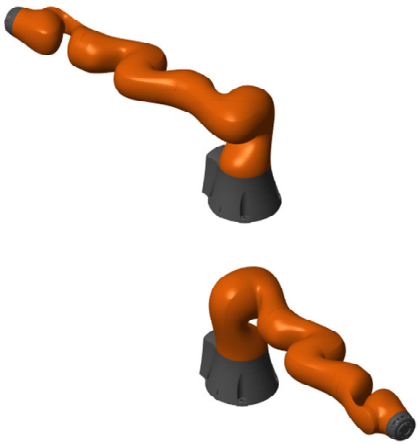

Figure 3.5: The single-joint sweeping motion.

To generate our trapenomial trajectory, the baseline trapezoid trajectory was used as input to the optimization problem described in Section 3.2. The bounds on acceleration, velocity, and jerk were inferred from the baseline trajectory and enforced in the optimization model. The optimization problem was then solved, and the solution constituted the trapenomial trajectory. To generate the polynomial trajectory, the above mentioned optimization problem was solved, however bounds on velocity and acceleration were relaxed by 20%.

Results and discussion The test trajectories and the resulting torque profiles are given in Figure 3.6. The trapezoidal velocity profile has distinct acceleration, cruising, and deceleration phases. Its corresponding torque profile indicates that at the end of the acceleration phase the demanded torque drops to a small non-zero value, which is required to maintain the velocity in the cruising phase.

Regarding the trapenomial velocity profile, one observes that during the acceleration phase, the velocity follows a polynomial path whose magnitude goes beyond that of the trapezoidal, albeit it remains below the maximum velocity limit. In the corresponding torque plot, it is seen that the trapenomial trajectory's torque never violates the maximum torque limits in the acceleration/deceleration phase. In the

cruising phase, the trapenomial trajectory follows the flat constant-speed profile of the trapezoidal trajectory, respecting the physical limits of the robot.

When following the polynomial velocity profile, the robot undergoes higher velocities than the limit by a factor of 4.3%, which is expected, since the velocity bounds were relaxed by 20% in the trajectory planning phase, as mentioned earlier. Likewise, as the bounds on acceleration were relaxed, robot demands higher torque in the acceleration and deceleration phase. Note that, in the deceleration phase, the negative torque values hint at an opportunity to regenerate power.

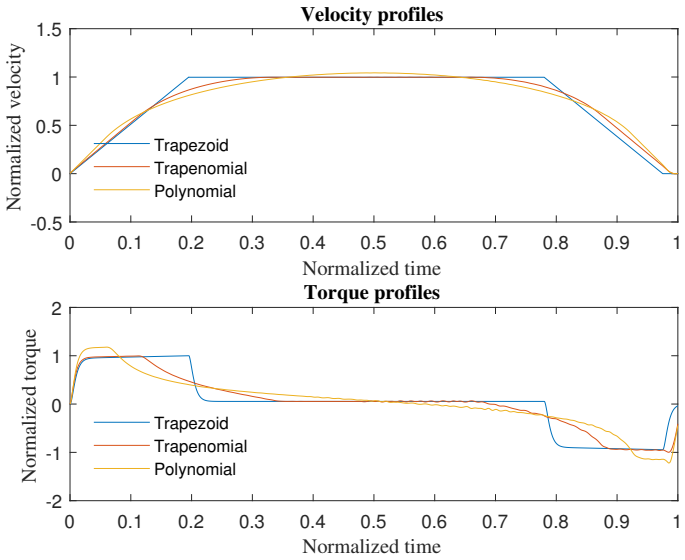

Figure 3.6: Different velocity and torque profiles vs. time.

Table 3.1 gives detailed information about energy consumption, violation of velocity and torque limits, and the performance criterion $\int \tau^2 dt$. As seen in the table, the polynomial trajectory yields the most reduction in the performance criterion $\int \tau^2 dt$, which amounts to less dissipated heat, which enables this trajectory to reduce energy consumption more than the trapenomial trajectory. However, this comes at the cost of violating the maximum permissible velocity and torque by 4.3% and 17.8%, which renders this trajectory unrealizable on a real physical robot.

On the other hand, following the trapenomial velocity profile results in a 15.6% reduction in the performance criterion $\int \tau^2 dt$, which amounts to 14.3% reduction in energy consumption, while no velocity or torque limits are violated. This makes the trapenomial velocity profile a viable energy-efficient alternative to its traditional trapezoidal counterpart. In terms of ease of implementation of such a trajectory on existing robots, we have proven in Paper 1 and 2 that this can be easily and successfully achieved on KUKA robots using the External Motion Interface (EMIly) built-in to KUKA System Software (KSS). Moreover, we have recently implemented the same concept on KUKA robots running KUKA Sunrise operating system using Fast Research Interface (FRI) library [68]. Our optimization technique not only aims to minimize energy consumption via more efficient velocity profiles, but takes advantage of waiting times in real-world trajectories to scale down velocity and acceleration, i.e. slowing down, which is a proven method to reduce energy consumption.

Table 3.1: Results of analysis of trapenomial and polynomial velocity profiles compared to the base-line trapezoidal trajectory; Reduction in energy consumption *Energy ↓ [%]*, percentage of violation of the joint's velocity limit *Velocity limit ↑ [%]*, reduction in integral of square of torque for the joint $\int \tau^2 dt \downarrow$ *[%]*, and percentage of violation of maximum allowable torque *Torque limit ↑ [%]*.

Velocity profile	Energy ↓ [%]	Velocity limit ↑ [%]	$\int \tau^2 dt \downarrow$ [%]	Torque limit ↑ [%]
Trapezoid	-	-	-	-
Trapenomial	14.3	0.0	15.6	0.0
Polynomial	23.2	4.3	17.2	17.8

3.4 Summary

In this chapter we provided supplementary information to our articles Paper 1 and Paper 2. More specifically, we explained our energy optimization procedure and how it compares with a traditional trapezoidal trajectory. A simulation with a simple single-joint motion was used to depict the effect of optimization, as well as an analogy with cars to further explain the concept.

Answering the research questions In Section 1.3 we formulated two research question, which are revisited, as follows.

RQ1: How can robot's energy consumption be reduced in a non-intrusive way, while preserving original path and cycle time?

RQ2: In what ways can energy and peak-power for multi-robot cell be reduced?

To answer *RQ1*, in Paper 1 we proposed a methodology for energy optimization of robots that preserves the path and cycle time that used the built-in robot functionalities without needing confidential parameters. This was also described earlier in this chapter.

To answer *RQ2*, in Paper 2, we refined our procedures further. We employed accurate energy measurement equipment and provided procedures to have reliable measurements and replicable experiments, along with a number of new modifications to our optimization model to target not only the energy consumption, but also the peak-power. Moreover, in Paper 2, the optimization method has been evaluated in many single and multi-robot scenarios, and on a variety of robots with different sizes. One key conclusion of the article is that, in addition to the trajectory optimization proposed in Paper 1, in multi-robot scenarios, it is possible to save energy even more. This is done by slowing down the robots that should wait for a resource to be freed. A detailed case study presented in Paper 2, which was conducted at Daimler, discussed this result.

Chapter 4

Routing and scheduling problems and algorithms

In this chapter, we aim to provide the reader with a background of the algorithms we used for routing and scheduling of moving devices, such as vehicles in a vehicle routing problem (VRP) context, and AGVs. Particularly, we will explain Benders decomposition, column generation, and gossip algorithms. We used Benders decomposition in Paper 5 and Paper 6, while column generation and gossip algorithms were used in Paper 3 and Paper 4. To this end, we start by a particular formulation of VRP as a toy example. Then Benders decomposition is discussed, which is followed by the gossip algorithm. Finally, column generation will be presented.

4.1 Vehicle routing problems

The classical VRP could be stated as the problem of determining an optimal set of routes for a set of vehicles such that given demands at the customers are satisfied. Each route starts and ends in a given depot. Each customer should be visited by one vehicle and each vehicle has a capacity that can not be exceeded. The routes are to be selected such that the total cost is minimized. This problem resembles the TSP, which was introduced in Chapter 2.

There are many different variations of the vehicle routing problem. For example, some variations may include single or multi-depot, heterogeneous capacities, time windows, sequencing constraints, and multiple objective functions, to name a few. For more information, the interested reader is referred to [69]. Next, we will present a particular variation called heterogeneous multi-vehicle routing problem (HMVRP), which will later be used to demonstrate how Benders decomposition and gossip algorithms work.

Heterogeneous multi-vehicle routing problem The formulation of HMVRP was first introduced in [70]. The term *heterogeneous* indicates that the vehicles are not alike, in contrast to VRP, where the vehicles all have the same capacity, speed, etc. In this section, we first present the formulation, and then consider a small HMVRP instance that will be used throughout the next sections to explain the

coming concepts. To stay consistent with the context in which Franceschelli et al. in [70] formulated the problem, the notion of robots and tasks will be used instead of vehicles and customers. To state this NP-hard problem, we consider the two following two sets.

- $\mathcal{N}$: The set of n robots
- $\mathcal{K}$: The set of k tasks

The movement speed and tasks execution speed of each robot may be different from another. Note that the movement speed refers to the speed at which a robot moves from one task to another, while the task execution speed refers to the speed at which a robot performs a task at a node that it must visit. Moreover, costs of the tasks vary depending on robots. To formulate the tours taken by robots we introduce the following sets:

- $\mathcal{V} = \mathcal{N} \cup \mathcal{K}$: The set of all nodes, i.e. $|\mathcal{V}| = n + k$.
- $\mathcal{E} = (\mathcal{N} \cup \mathcal{K}) \times (\mathcal{N} \cup \mathcal{K})$: The set of all directed edges, i.e. $|\mathcal{E}| = (n + k)^2$.

To formalize a MILP model, two sets of binary variables are required; one for task assignment and the other for sequence planning, as follows

- $\mathcal{X}$: The set of all task assignment variables x_{ir}, where $i \in \mathcal{V}$ and $r \in \mathcal{N}$ with $|\mathcal{X}| = n \times (n + k)$. Also, if $i \in \mathcal{N}$ and $x_{ir} = 1$: Robot R_r starts its tour from depot (node) i, and if $i \in \mathcal{K}$ and $x_{ir} = 1$: Task i is executed by robot R_r.
- $\mathcal{Y}$: The set of all sequencing variables y_{ijr}, where $(i, j) \in \mathcal{E}$ and $r \in \mathcal{N}$ with $|\mathcal{Y}| = n \times (n + k)^2$. Notice that $y_{ijr} = 1$ means that robot R_r goes directly from node i to node j. To prevent from taking self-loops, large costs should be assigned to edges with $i = j$.

Additionally, a continuous variable λ is defined to model the makespan. Finally, the following costs are defined:

- c_{ir}: The execution time of task i with the cost c_i ($i \in \mathcal{K}$) by robot R_r ($r \in \mathcal{N}$) with an execution speed w_r, i.e. $c_{ir} = c_i/w_r$.
- d_{ijr}: Represents the time spent by robot R_r to travel the Euclidean distance along the edge $(i, j) \in \mathcal{E}$ with speed v_r.

Ultimately, the *centralized* MILP formulation is as follows:

$$\min \quad \lambda \tag{4.1}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{K}} x_{ir} c_{ir} + \sum_{(i,j) \in \mathcal{E}} d_{ijr} y_{ijr} < \lambda \qquad \forall r \in \mathcal{N} \tag{4.2}$$

$$x_{rr} = 1 \qquad \forall r \in \mathcal{N} \tag{4.3}$$

$$\sum_{r \in \mathcal{N}} x_{ir} = 1 \qquad \forall i \in \mathcal{K} \tag{4.4}$$

$$\sum_{j \in \mathcal{V}} y_{ijr} = \sum_{j \in \mathcal{V}} y_{jir} = x_{ir} \qquad \forall i \in \mathcal{V},\ \forall r \in \mathcal{N} \tag{4.5}$$

$$\sum_{i \notin \mathcal{S}} \sum_{j \in \mathcal{S}} y_{ijr} \geq x_{qr} \qquad \forall \mathcal{S} \subseteq \mathcal{K}, \forall q \in \mathcal{S}, \forall r \in \mathcal{N} \tag{4.6}$$

$$\lambda \in \mathbb{R} \tag{4.7}$$

$$x_{ir} \in \{0, 1\} \qquad \forall i \in \mathcal{V},\ \forall r \in \mathcal{N} \tag{4.8}$$

$$y_{ijr} \in \{0, 1\} \qquad \forall (i, j) \in \mathcal{E},\ \forall r \in \mathcal{N} \tag{4.9}$$

The solution to the above problem yields the optimal task assignment [70]. The constraints in (4.2) along with the objective function (4.1) aim to minimize the maximum execution time of robots (makespan). The constraints in (4.3) ensure that each robot is dispatched from its depot. The constraints in (4.4) guarantee the assignment of each task to just one robot. The constraints in (4.5) mean that exactly one edge enters and exactly one edge leaves a node (depot or task), when the node is assigned to a robot. Furthermore, the constraints in (4.6) represent the sub-tour elimination constraints (see Section 2.2 for more information on SECc). Finally, the constraints in (4.7), (4.8), and (4.9) define the variables' domains.

Example: A small HMVRP instance Consider two robots labeled A and B that are to be assigned to 5 tasks numbered from 1 to 5. The layout of the robots and tasks is shown in Figure 4.1 (left). The optimal solution to this problem can be seen in the same figure (right).

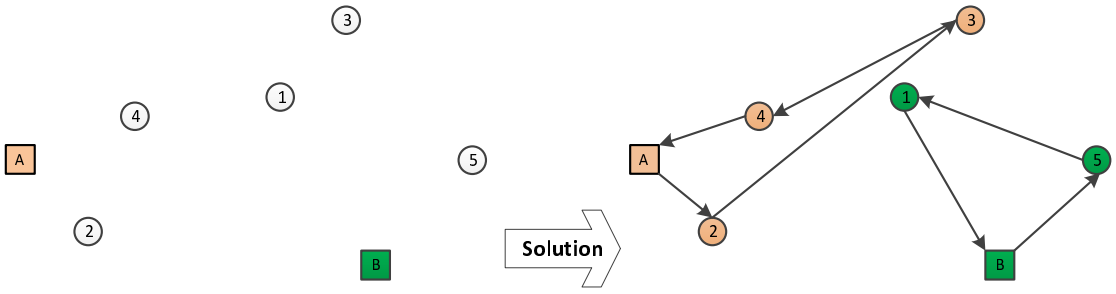

Figure 4.1: Left: Problem layout Right: Optimal solution

We can see that the tasks assigned to a particular robot are indicated in the same color as the robot itself. The arrows show the sequence in which the tasks are executed. Indicating the optimal makespan of robots A and B by λ_A and λ_B, and

the optimal makespan by λ^*, the following values are given by the optimal solution:

$$\lambda_A = 13.05,\ \lambda_B = 13.44,\ \lambda^* = 13.44$$

It means that the total makespan is lower bounded by the makespan of the slowest robot, namely robot B. Another observation is that for each related set of robot-tasks (indicated in the same color), the robot simply takes the shortest possible rout. This is equal to solving a traveling salesman problem (TSP) for each individual robot, provided that we know which tasks are assigned to it. This observation leads to utilizing a method that decomposes the HMVRP into a *task assignment* problem, and a number of *TSPs* as *sequencing* problems. One such method is called Logic based Benders decomposition that we will explain next.

4.2 Benders decomposition

The Benders decomposition technique is a constraint-directed search method that was first introduced in the early 60s by Benders [71]. Benders' original work is today referred to as the *classical* Benders decomposition, as opposed to the *logic-based* Benders decomposition that first appeared in the mid-90s.

The idea of the Benders decomposition is to take advantage of the problem structure to facilitate the solution procedure. This is done by partitioning the problem into easier-to-handle segments. As an example, consider a MILP with a mixture of continuous (easy) and integer (difficult) variables. To decompose the MILP, we construct an auxiliary problem by selecting the constraints including only the difficult variables. This auxiliary problem can be considered a relaxation to the original MILP, and is formally called the *master* problem. Next, the solution of the master problem is used to construct a second auxiliary problem, by fixing the difficult variables in the MILP to the solution values. This forms another problem called the *slave* problem (or sub-problem), which is now easier to solve due to lack of difficult variables, since it is an LP.

A third notion in Benders context is the concept of *cuts*. In the classical Benders method, the solutions of the dual sub-problems is used to generate a special constraint called the cut. The cut is then added to the master problem. The master problem is now re-optimized, and this process is repeated. The objective function value of the slave problem yields an upper bound for the main problem, while solving the master problem generates a lower bound. The optimality is achieved when the upper and lower bounds meet.

The classic Benders method is a special case of logic-based Benders decomposition (LBBD). In LBBD, sub-problems need not to be linear programming and can take any form. A cut is obtained from each sub-problem by an inference method, and in particular, by solving an inference dual of the sub-problem. For each category of sub-problems, a separate analysis should be carried out to determine the cuts. So, unlike the classic cuts, no general formulation for LBBD cuts is available.

Generic Benders decomposition To understand Benders method, we first recall from Section 2.5 how a generic constraint-directed search algorithm works. To

facilitate the search, a series of problem restrictions $P(x^1), ...P(x^m)$ are solved. A problem restriction $P(x^k)$ is obtained by fixing all or a subset of variables to their respective solutions, i.e. $(x_1^k, ..., x_n^k) = (v_1^k, ..., v_n^k)$, where $v_i^k \in D$. To find the solution x^k, first the set of nogoods $\mathcal{N}_k$ (also known as relaxation R_k) must be solved. Next, problem restriction $P(x^{k+1}) = P(x^k)$ is defined and its optimal value is used to generate nogood N_{k+1}. Then the nogood set is updated $\mathcal{N}_{k+1} = \mathcal{N}_k \cup N_{k+1}$, and the process is repeated.

In Benders context, a nogood bound is referred to as Benders cut, the set of nogood bounds R_k is called *master problem*, and restriction $P(x)$ is called the subproblem. In each iteration, always the same subset of variables are fixed to their solution values to create a new subproblem, i.e. when a solution x^k to the master problem is obtained, to form the next subproblem we fix $(x_1^k, ..., x_p^k)$ to $(v_1^k, ..., v_p^k)$, and in every iteration we fix exactly the same variables, however to different values (notice that $\{x_1, ..., x_p\} \subset \{x_1, ..., x_n\}$). The key question is that which subset of variables should be fixed to form subproblems for which efficient algorithms are known?

To cast light on the importance of correct variable selection, consider a mixed integer linear programming problem (MILP) with a mixture of continuous (easy) and integer (difficult) variables as follows:

$$\begin{aligned} \min\ & cx + fy \\ \text{s.t. } & Ax + By \geq b \\ & x \in \mathbb{Z}^m \\ & y \in \mathbb{R}^n \end{aligned}$$

Notice that $x \in \mathbb{Z}^n$ is the vector of complicating variables. To create a restriction (subproblem) that is easy to solve, one can fix x to some trial value $\bar{x} \in \mathbb{Z}^m$. The resulting subproblem will be a linear programming problem as below, which can be efficiently solved by the simplex algorithm.

$$\begin{aligned} \min\ & c\bar{x} + fy \\ \text{s.t. } & By \geq b - A\bar{x} \\ & y \in \mathbb{R}^n \end{aligned}$$

In Benders context, the variables to be fixed are referred to as search variables, and the rest, as subproblem variables. Hence, the first step in Benders method is to partition the variables into two vectors; vector of search variables x, and vector of subproblem variables y. Note that in general, both the search and subproblem variables can be difficult variables, but partitioning the problem could still lead to subproblems for which efficient algorithms exists. Having said that, a general minimization problem P can now be written as

$$\begin{aligned} & \min\ f(x, y) \\ & \mathcal{S}(x, y) \\ & \mathcal{S}(x), \mathcal{S}(y) \\ & x \in D_x, y \in D_y \end{aligned} \tag{4.10}$$

In problem (4.10), $\mathcal{S}(x, y)$ is the constraint set involving both x and y, while $\mathcal{S}(x)$ and $\mathcal{S}(y)$ include only constraints involving x and y respectively.

The next step in Benders method is to decompose (4.10) into a master problem, and one (or more) subproblem(s). The master problem must involve only the search variables, and subproblems only the subproblem variables.

As a constraint-directed method, in iteration k of Benders, relaxation R_k (the master problem) should be solved to obtain a solution that will define the next restriction (subproblem). Recall that in constraint-directed search, the relaxation is written as in Problem (2.8). The nogood bounds (Benders cuts) must be formulated using only search variables x. Additionally, any other constraint involving *only* x is added to the relaxation. Together, they form the master problem, as follows.

$$\begin{aligned} &\min\ v \\ &v \geq B_i(x),\ i = 1, ..., k \\ &\mathcal{S}(x) \\ &x \in D_x \end{aligned} \tag{4.11}$$

Note that the master problem is a relaxation that gives a lower bound on v, since the restrictions $\mathcal{S}(x, y)$ and $\mathcal{S}(y)$ are not included.

Now that we have the master problem in step k, we should solve it to obtain the solution x^k. Then, the problem restriction (subproblem) $P(x^k)$ becomes as follows:

$$\begin{aligned} &\min\ f(x^k, y) \\ &\mathcal{S}(x^k, y) \\ &\mathcal{S}(y) \\ &y \in D_y \end{aligned} \tag{4.12}$$

It is clear that the search variables are not present in the subproblem (4.12). Observe that the constraints in $\mathcal{S}(x^k, y)$ now only involve y, and that is because x variables have been fixed to their solution values x^k.

In a generic constraint-directed method, solving a restriction $P(x^k)$ yields the optimal value pertaining to that restriction, i.e. $v(x^k)$. Then a nogood is generated and added to the nogood set (relaxation). Similarly in Benders, when the subproblem is solved, a Benders cut is generated and added to the master problem to form the next master problem. As mentioned earlier, the Benders cuts are formulated using x (that is $v \geq B_{k+1}(x)$).This is because the master problem was originally formulated using x to yield a solution x^k that defines $P(x^k)$.

Benders algorithm closely mimics the generic constraint-directed search (Algorithm 1) and is given below:

The algorithm terminates when the lower bound meets the upper bound, that is when $v_{LB} = v_{UB}$.

A Benders decomposition for HMVRP To clarify the Benders method, we consider its application to the heterogeneous multi-vehicle routing problem (HMVRP) as a case study, which was already introduced in Section 4.1. Decomposing

Let $v_{LB} = -\infty$ and $v_{UB} = \infty$.
Initially master problem (4.11) minimizes v subject to $x \in D_x$ and possibly other valid constraints involving x.
Let $\bar{x}$ be a feasible solution of (4.11).
while $v_{LB} < v_{UB}$ **do**
  Solve the subproblem (4.12) and let $v(\bar{x})$ be the minimum value of $f(\bar{x}, y)$ subject to $\mathcal{S}(\bar{x}, y)$ and $y \in D_y$.
  Let $v_{UB} = \min\{v(\bar{x}), v_{UB}\}$.
  Add a Benders cut $v \geq B(x)$ to the master problem (4.11).
  Solve the master problem (4.11) and obtain its optimal value v_{LB}.
end
The optimal value of P is v_{UB} ($v = v_{UB}$).

Algorithm 2: Generic Benders algorithm for minimizing $f(x, y)$ subject to $\mathcal{S}(x, y)$, $\mathcal{S}(x)$, $\mathcal{S}(y)$ and $(x, y) \in D_x \times D_y$ [34].

the HMVRP results in a master problem in form of mixed integer linear programming (MILP), and a cluster of MILP subproblems resembling TSPs.

Master problem: the task assignment The role of the master problem for HMVRP is to find a solution to the task assignment problem. Hence, the family of constraints solely composed of x_{ir} (constraints (4.3), (4.4) and (4.8)) are included there. Other constraints to add are the Benders cuts, and relaxations of the subproblems, if available. The relaxations are, in effect, constraints that prevent from getting infeasible subproblmes. The solution of the master problem results in generating a set of sequencing TSP subproblems. As mentioned earlier, the objective function in the centralized formulation comprises minimizing the maximum makespan λ. So, the compatible master problem structure is chosen from [34]. This leads to the following MILP:

$$\min \quad z \tag{4.13}$$

$$\text{s.t.} \quad x_{rr} = 1 \qquad \forall r \in \mathcal{N} \tag{4.14}$$

$$\sum_{r \in \mathcal{N}} x_{ir} = 1 \qquad \forall i \in \mathcal{K} \tag{4.15}$$

$$x_{ir} \in \{0, 1\} \qquad \forall i \in \mathcal{V},\ \forall r \in \mathcal{N} \tag{4.16}$$

$$Benders\ Cuts \tag{4.17}$$

$$Optional\ relaxations \tag{4.18}$$

Notice that z is an auxiliary continuous variable acting as a substitute for λ, and its optimal value in each iteration yields a lower bound for the makespan.

Subproblems: the sequencing TSPs To formulate each subproblem, one robot and the set of its allocated tasks are considered. Note that our HMVRP is assumed to be collision-free, so any assignment to the master problem leads to a

number of fully decoupled subproblems. A subproblem corresponding to robot R_r is denoted by $\mathbb{S}_r$. Suppose that the solution set of the master problem in the l^{th} iteration is denoted by $\mathcal{X}^l$. Then, the following notations for $\mathbb{S}_r$ are defined:

- $\mathcal{N}_r$: The single-member set of the depot node of R_r.
- $\mathcal{K}_r^l$: The set of tasks assigned to robot R_r.
- $\mathcal{V}_r^l = \mathcal{N}_r \cup \mathcal{K}_r^l$: The set of all nodes.
- $\mathcal{E}_r^l = \mathcal{V}_r^l \times \mathcal{V}_r^l$: The set of directed edges.
- $x_{ir}^l = 1$: Solution to the assignment variable x_{ir} in the l^{th} iteration, where $(x_{ir}^l \in \mathcal{X}^l)$, $(i \in \mathcal{K}_r^l), (r \in \mathcal{N}_r)$; i.e. the task i will be performed by R_r.

Then, the single TSP in $\mathbb{S}_r$ can be formulated as:

$$\min \quad \lambda_r^l \tag{4.19}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{K}_r^l} x_{ir}^l c_{ir} + \sum_{(i,j) \in \mathcal{E}_r^l} d_{ijr} y_{ijr} < \lambda_r^l \qquad \forall r \in \mathcal{N}_r \tag{4.20}$$

$$\sum_{j \in \mathcal{V}_r^l} y_{ijr} = \sum_{j \in \mathcal{V}_r^l} y_{jir} = x_{ir}^l \qquad \forall i \in \mathcal{V}_r^l,\ \forall r \in \mathcal{N}_r \tag{4.21}$$

$$\sum_{i \notin \mathcal{S}} \sum_{j \in \mathcal{S}} y_{ijr} \geq x_{qr}^l \qquad \forall \mathcal{S} \subseteq \mathcal{K}_r^l,\ \forall q \in \mathcal{S},\ \forall r \in \mathcal{N}_r \tag{4.22}$$

$$\lambda_r^l \in \mathbb{R} \tag{4.23}$$

$$y_{ijr} \in \{0, 1\} \qquad \forall r \in \mathcal{N}_r,\ \forall (i,j) \in \mathcal{E}_r^l \tag{4.24}$$

Notice that in constraint (4.20), the first left-hand side term is now a constant, and the same is true for the right-hand sides of (4.21) and (4.22). The subproblem now resembles the classic asymmetric TSP for which an extensive body of research exists.

Benders cuts for HMVRP The Benders cuts are arguably the most important ingredient of the Benders decomposition algorithm. The cuts are generated using the solution of the subproblems, and then added to the master problem to direct the search. Notice that the cuts become available to the master problem *after* the first iteration, when the early solutions of the subproblems have been obtained.

Now, a simple Benders cut, which is frequently used in planning and scheduling problems, is formulated as in [34]. To state the formulation, define $\mathcal{J}_r$ as the set of tasks assigned to robot R_r, and the cut becomes:

$$z \geq \lambda_r^* (\sum_{i \in \mathcal{K}_r} x_{ir} - |\mathcal{J}_r| + 1)$$

where λ_r^* is the optimal objective function value of the r^{th} subproblem, and $|\mathcal{J}_r|$ is the number of tasks assigned to robot R_r. It means that if exactly the same set of tasks are assigned to machine R_r, the makespan of this particular subproblem will be at least λ_r^*. So, in order to obtain a shorter makespan, the solver should avoid allocating the same set of tasks to robot R_r.

How the cuts work Consider again the small HMVRP example in Section 4.1. Assume that in some iteration of the Benders algorithm, the solution to the master problem assigns tasks 1, 4 and 5 on robot A, and tasks 2 and 3 on robot B, that is $x_{A1} = x_{A4} = x_{A5} = 1$, and $x_{B2} = x_{B3} = 1$, while other assignment variables become zero. This assignment is depicted in Figure 4.2 (left). Solving the resulting subproblems gives makespans $\lambda_A = 13.98, \lambda_B = 16.48$ for robots A and B respectively. Since the the total makespan is bound to the makespan of the slowest robot, for this particular iteration, the overall makespan is 16.48. We also know from previous iterations that the best makespan so far (incumbent solution) has been $\lambda^* = 14.84$. It is clear that the current assignment is undesirable, and we need to avoid this and *similar* assignments in the future search. To understand what we mean by similar assignment, notice that adding another task to robot B makes this robot take a longer trip, indicating an increase in its makespan, which will definitely go beyond 16.98 units of time, see Figure 4.2 (right). Such an assignment is undesirable ($\lambda > 16.98 > \lambda^* = 14.84$) and should be avoided.

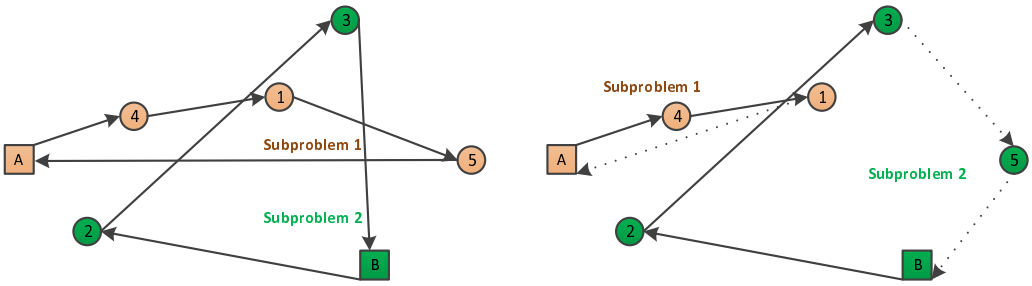

Figure 4.2: (Left) Solution to an iteration of the small example. (Right) A similar tasks assignment on robot B with more tasks worsens the makespan.

The mechanism behind avoiding bad assignments is as follows: The search procedure (ex. a search tree) in the master problem records the current best solution (incumbent solution). Before testing a new assignment, it tries to use knowledge obtained in previous steps of the search to determine whether the next assignment is going to yield a solution that is worse than the current incumbent solution. This knowledge has already been recorded in the master problem in form of the Benders cuts. If the available cuts contain enough knowledge to flag a potential solution undesirable, that assignment will be skipped. Otherwise, the current candidate assignment should be tried to compute its corresponding makespan value(s) from the subproblem(s).

In this example, to generate cuts that exclude the current and similar assignments with worse maskespans, one can formulate the following cuts:

$$\lambda \geq 13.98 * (x_{A1} + x_{A4} + x_{A5} - 2)$$
$$\lambda \geq 16.48(x_{B2} + x_{B3} - 1)$$

Currently, the second cut (pertaining to robot B) is the one that does the job in

preventing some bad assignments, as follows:

$x_{B2} = x_{B3} = x_{B1} = 1,$
$x_{B2} = x_{B3} = x_{B4} = 1,$
$x_{B2} = x_{B3} = x_{B5} = 1,$
$x_{B2} = x_{B3} = x_{B1} = x_{B4} = 1,$
$x_{B2} = x_{B3} = x_{B1} = x_{B5} = 1,$
$x_{B2} = x_{B3} = x_{B4} = x_{B5} = 1$

To study the cut further, three scenarios can be imagined:

i) **The same assignments:** In this case the cuts simply yield $\lambda \geq 13.98$, and $\lambda \geq 16.48$, which is what we expect (recall the general nogood formulation (2.7) from Section 2.5).

ii) **Adding tasks:** For instance, having the assignment $x_{B2} = x_{B3} = x_{B1} = 1$ activates the second cut as $\lambda \geq 16.48$, and this will result in this assignment to be skipped in the search tree, because the search procedure has flagged this assignment as something that will yield a worse solution than the current incumbent, i.e. $\lambda = 16.48 > \lambda^* = 14.84$.

iii) **Removing tasks:** taking one or more task from a robot decreases its makespan. That means in nogood bound (2.7), $x \notin T$, and $B_{k+1}(x) \to -\infty$. However, it is not necessary to go to $-\infty$, rather, a small enough lower bound suffices, as it happens here. For example, when task 2 is removed from robot B, we have $\lambda \geq 16.48 * (0 + 1 - 1)$, or simply $\lambda \geq 0$, which is a small enough valid lower bound.

One important remark is that, sometimes cuts that have been ineffective so far can become active and effective in the future. For example, the first cut in the above example (pertaining to robot A) does nothing at the moment, since it can not guarantee that adding tasks to robot A will increase its makespan up to a value more than the current incumbent, i.e. all we can say about an assignment like $x_{A1} = x_{A4} = x_{A5} = x_{A2} = 1$ is that $\lambda \geq 13.98$, and this is allowed in the search procedure, but we do not know whether $\lambda \geq 14.84$. However, in one of the future iterations, the search procedure in the master problem will find a new incumbent as $\lambda^* = 13.44$. This will make the first cut prevent these assignments:

$x_{A1} = x_{A4} = x_{A5} = x_{A2} = 1,$
$x_{A1} = x_{A4} = x_{A5} = x_{A3} = 1,$
$x_{A1} = x_{A4} = x_{A5} = x_{A2} = x_{A3} = 1$

All of the potential assignment above result in the first cut to become $\lambda = 13.98 > \lambda^* = 13.44$, and this will flag those assignments as undesirable, so they will be skipped.

How the algorithm progresses For the same small example, applying Benders decomposition results in a master problem and two subproblems, one per each robot. In each iteration of the master problem, some tasks are assigned to each robot. Figure 4.3 shows the task assignment for the iteration 7 of the problem. Every time a subproblem is solved, the tasks in that subproblem are scheduled on the pertaining robot. This can be seen, for step 7, in Figure 4.4.

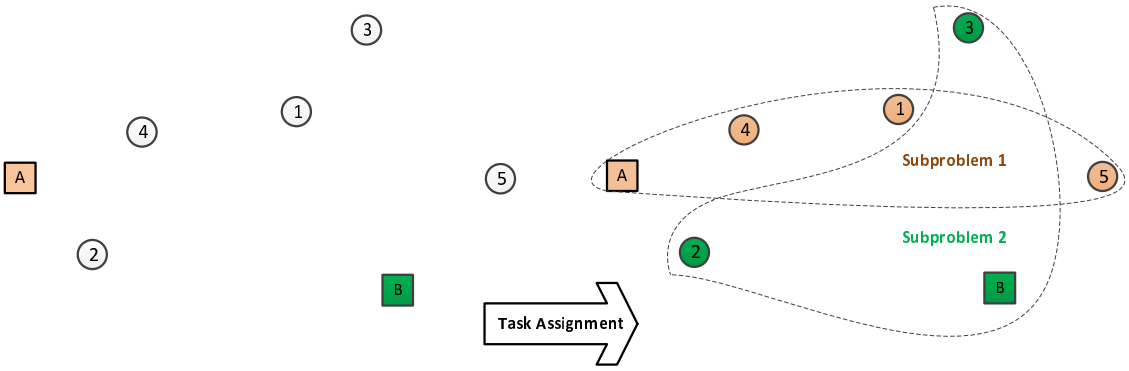

Figure 4.3: Master problem: the task assignment.

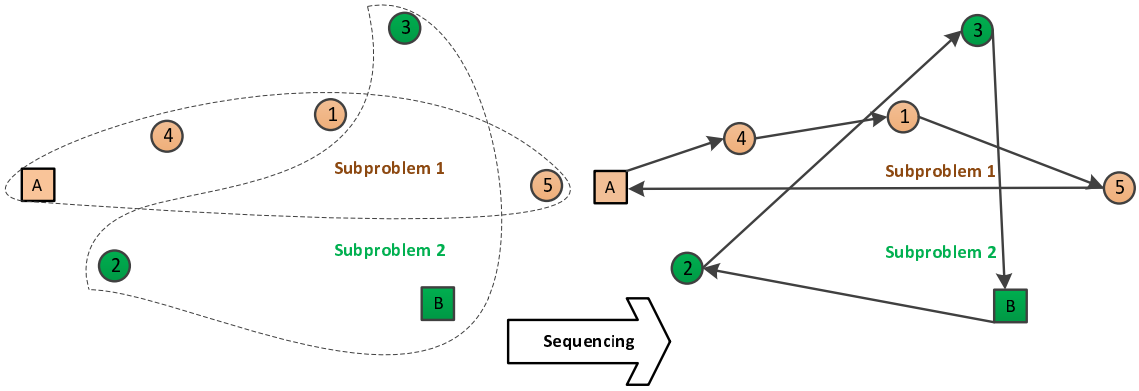

Figure 4.4: Slave problem: the sequencing.

Now we take a look at a few iterations of the algorithm that are summarized in Table 4.1. (Note that before the branching tree for the master problem is constructed, CPLEX heuristically finds a few solutions, in this case, the first 4 solutions in the table, using its *pre-solve* procedure).

According to the table, the optimal makespan is $\lambda = 13.44$, that happens in iterations 2 and 10. As said before, the first 4 rows of the table were generated using the pre-solve procedure in CPLEX, and the rest are obtained from the branch and bound search tree that is generated after the pre-solve. This is why that particular solution has happened twice.

Iter.	R_r	Assignment	Sequence	makespan
1	R_1	T_1, T_3, T_4	T_1, T_3, T_4	11.48
	R_2	T_2, T_5	T_2, T_5	15.71
2	R_1	T_2, T_3, T_4	T_2, T_3, T_4	13.05
	R_2	T_1, T_5	T_1, T_5	13.44
3	R_1	T_1, T_2, T_4	T_1, T_2, T_4	10.57
	R_2	T_3, T_5	T_3, T_5	13.99
4	R_1	T_1, T_2, T_3, T_5	T_2, T_5, T_3, T_1	19.25
	R_2	T_4	T_4	9.71
5	R_1	T_1, T_2, T_3, T_4	T_2, T_1, T_3, T_4	14.84
	R_2	T_5	T_5	7.94
6	R_1	T_2, T_4, T_5	T_4, T_5, T_2	15.20
	R_2	T_1, T_3	T_1, T_3	13.41
7	R_1	T_1, T_4, T_5	T_4, T_1, T_5	13.98
	R_2	T_2, T_3	T_2, T_3	16.48
8	R_1	T_3, T_4, T_5	T_5, T_3, T_4	15.09
	R_2	T_1, T_2	T_2, T_1	14.47
9	R_1	T_2, T_5	T_2, T_5	13.95
	R_2	T_1, T_3, T_4	T_4, T_3, T_1	15.92
10	R_1	T_2, T_3, T_4	T_2, T_3, T_4	13.05
	R_2	T_1, T_5	T_1, T_5	13.44
11	R_1	T_1, T_2, T_3	T_2, T_3, T_1	14.05
	R_2	T_4, T_5	T_4, T_5	14.27
12	R_1	T_1, T_3, T_5	T_1, T_3, T_5	16.11
	R_2	T_2, T_4	T_2, T_4	12.81

Table 4.1: Results of the different iterations: The tasks assignments, sequences, and makespans

4.3 Column generation

The column generation (CG) is another famous decomposition algorithm [39], whose relationship with Benders decomposition is orthogonal. In each Benders decomposition, a number of constraints are added to the problem, that is, the problem grows vertically, as more rows are added to its formulation. In CG, however, the problem is decomposed in a way that, in each iteration one or several columns (variables) are added to the problem, hence the name column generation. Thus, the problem formulation grows horizontally, while the number of rows (constraints) is fixed.

At the core of the CG algorithms there is a decomposition mechanism known as the Dantzig-Wolfe decomposition (DWD) [38]. The idea behind a decomposition approach is to take advantage of the problem structure to speed up the solution procedure. This is achieved by partitioning the problem into smaller and more manageable subproblems that are solved separately, and by letting them communicate with each other. DWD is one such method that was first developed for linear programming (LP) problems with block-angular constraint matrix.

Consider an LP with a combination of easy constraints and complicating (coupling) ones. In the coupling constraints some variables are coupled together with non-zero coefficients. After identifying the complicating constraints the problem is *reformulated* such that a feasible solution is represented by a convex combination of the extreme points of the polyhedron defined by the easy constraints. The reformulated problem is called the *master problem* (MP), and it has fewer constraints but considerably more variables. Due to the large number of variables, this new problem is difficult to construct, and to solve by simplex algorithm. Since in every iteration of simplex algorithm only a few of the variables are present in the basis, one may choose to construct and start the master problem based on only a meaningful subset of the extreme points (variables). This is known as the *restricted master problem* (RMP). To improve the objective function value of the RMP, a new variable should enter the basis. The new variable can be identified by solving one (or more) auxiliary problem(s) known as the *subproblem(s)*. In other words, any variable in the RMP corresponds to a solution to one of the subproblems.

A subproblem is defined as the problem of finding a variable that improves the objective function of the RMP the most, such that the easy constraints are satisfied. Hence, the objective function value of the subproblem is the *reduced cost* of a variable entering the basis. The communication of the RMP with the subproblem is done through manipulating the objective function of the subproblem by using solution values of the dual variables corresponding to the RMP. In case of a minimization problem, if the reduced cost is negative, a new variable (column) is added to the RMP together with its corresponding coefficients. This process continues until no new variable can be found. Hence, instead of solving the original problem, a restricted master problem is solved together with one or more subproblem(s), while these smaller problems exchange information. It is possible to apply DWD to structured MILP problems as well, as in [39]. For a more detailed treatment of the topic, see [39].

4.4 Gossip algorithm

The gossip algorithm [72], is used in telecommunication and sensor networks to exchange information and for computations in network of nodes. In the algorithm, every node in the network communicates with its neighboring node, hence the computations are done in a decentralized manner.

In a vehicle routing problem context, according to the gossip rule, after initial task assignment, two vehicles and their corresponding tasks are picked randomly and a local optimization problem is generated. Either an equal or a better objective function value can be obtained by solving this problem. This is achieved by retaining or exchanging the tasks between the vehicles, and the procedure is then repeated.

Now we give an illustrative example. Consider a small instance of an HMVRP with 3 vehicles and 9 tasks, as shown in Figure 4.5. Each set of vehicle-tasks is shown in the same color. In the first iteration (upper-left corner), an arbitrary feasible task assignment leads to (tasks 1, 2, 3 → vehicle A), (tasks 4, 5, 8 → vehicle B), (tasks 6, 7, 9 → vehicle C). Then, two vehicles and their tasks are chosen randomly. For example, vehicle A and B, and a local optimization problem is generated. The solution to the new problem results in task 4 to be passed to vehicle A, as this reduces the objective function value (distance) of the local problem. Next, the vehicles B and C are chosen, and task 9 is transferred from C to B. Then, vehicles A and C are picked, without any improvement, as they are already optimal in the local problem. The idea behind the algorithm is that it allows to solve smaller problems compared to a monolithic (centralized) approach.

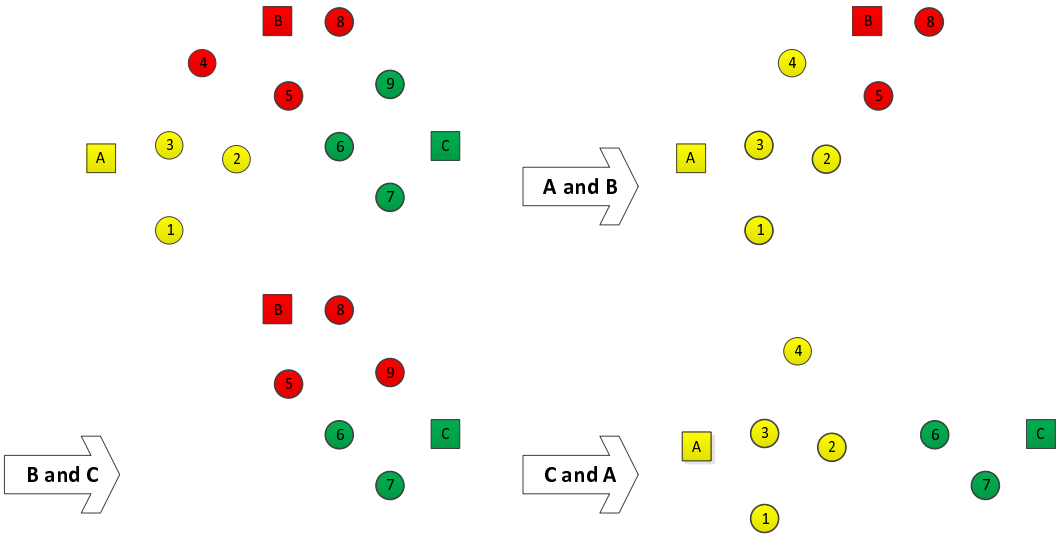

Figure 4.5: Some iterations of gossip algorithm for a small instance of HMVRP.

Note that the gossip algorithm has similarities to the POPMUSIC (Partial Optimization Metaheuristic under Special Intensification Conditions) proposed by Taillard and Voss [73], but it is different in the way the local problems are formed. In POPMUSIC, a local problem is created by randomly selecting a seed tour and

then considering an independent VRP containing only the customers of the r closest tours of the seed tour, while r is automatically adjusted. In other words, except for the first tour in the local problem, the other tours that are to be included are selected using a selection procedure. A selection criterion is for example that the centroid of all tours should be close to each other. In the gossip framework, however, all vehicles together with their corresponding tours are chosen randomly to form the subproblems.

The gossip algorithm is described in Algorithm 3. In this algorithm, after setting the limit on the iterations, a feasible initial solution must be generated. It determines the initial size of the pool of the vehicles, tours for each vehicle, and the length of each tour. In the main loop of the algorithm, a local problem consisting of two vehicles and their tours is defined and then solved using any solver. The result of the local optimization is then reported back. If a vehicle has now an empty tour, it will be removed from the pool of the existing vehicles. If the new tour of the vehicle is shorter than the old one from the previous iteration, the old tour and its length are replaced with the new one. The algorithm continues until the maximum number of iterations (or a time limit) is reached.

```
Initialize IterLim;
nbVehicles, tours, tourLengths ← InitialSolution();
Set counter to 0;
while counter ≤ IterLim do
    Randomly pick two distinct vehicles;
    Form a local VRP comprising the two vehicles and their tours;
    Optimize the local problem;
    for vehicles in the local optimization do
        if the tour is empty then
            nbVehicles ← nbVehicles − 1;
        end
        if the tour is shorter then
            Update(tours, tourLengts);
        end
    end
    counter ← counter + 1
end
Report sum(tourLength)
```

Algorithm 3: Gossip algorithm for a generic vehicle routing problem

Parallel gossip In Paper 3, the gossip algorithm was applied to a variation of VRP with time windows, with promising results. Later, in Paper 4 we proposed a parallel version of the gossip algorithm with even better results. In that paper, we utilized the distributed nature of the gossip algorithm to solve the local problems in parallel. One feature of the method is that the required effort for parallelization is minimal, and that the whole framework can be used for different variations of VRP.

Algorithm 4 describes the multi-threaded version of Algorithm 3. In the first line, the number of threads and maximum number of iterations are set. Then, a feasible initial solution must be generated. It determines the initial size of the pool of the vehicles, tours for each vehicle, and the length of each tour. As noted in line 6 and 7, each thread will have a local problem consisting of two vehicles and their tours, such that each vehicle appears in only one thread. In the end of the parallelization phase, each thread reports back its optimization results. If a vehicle has now an empty tour, it will be removed from the pool of the existing vehicles. If the new tour of the vehicle is shorter than the old one from the previous iteration, the old tour and its length are replaced with the new one. The algorithm continues until the maximum number of iterations (or a time limit) is reached.

```
Initialize Threads, IterLim;
nbVehicles, tours, tourLengths ← InitialSolution();
Set counter to 0;
while counter ≤ IterLim do
    for thread in Threads do
        Randomly pick two distinct vehicles;
        Form local VRPs: assign to each thread two vehicles and their tours;
    end
    Beginning of parallelization ;
    for thread in Threads do
        Optimize tours of the two vehicles;
    end
    End of parallelization;
    for threads in Threads do
        for vehicles in the local optimization do
            if the tour is empty then
                nbVehicles ← nbVehicles − 1;
            end
            if the tour is shorter then
                Update(tours, tourLengts);
            end
        end
    end
    counter ← counter + 1
end
Report sum(tourLength)
```

Algorithm 4: Generic multi-threaded gossip algorithm

4.5 Case study: healthcare routing and scheduling

Home healthcare is about dispatching caregivers to people in need of healthcare service at home. The task assignment and route generation for caregivers can be formulated as an extension of the well-known vehicle routing problem with time windows (VRPTW).

Currently, the most successful exact algorithms for VRPTW are based on CG. While such methods could be successful for home healthcare routing and scheduling problems (HHCRSPs) as well, fast approximate algorithms are appealing, especially for large problems. In our case study, given in Paper 3, we combined CG and the gossip algorithm, which was called gossip-CG, to generate an algorithm that is more successful than CG alone. Figure 4.6 illustrates the architecture of the gossip-CG method, where a local problem consists of two caregivers and their corresponding nodes, optimized using column generation.

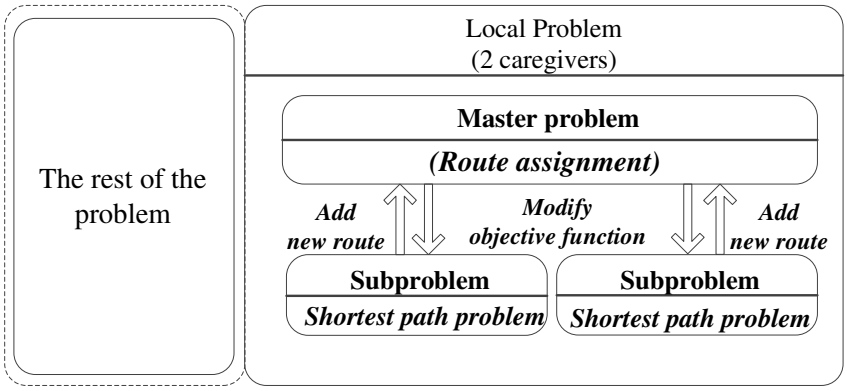

Figure 4.6: The architecture of the gossip-CG method. A local problem consists of two caregivers and their corresponding nodes, optimized using column generation independent of the rest of the problem.

Optimization model and algorithms The full optimization model, which is a variation of VRP, can be found in Paper 3, where the CG algorithm and its application to the problem is described in full detail.

The gossip algorithm can be used with a standard MILP solver for the local problems, for example using CPLEX. Then, the method is called gossip-CPLEX. Since the performance of a gossip algorithm depends on the performance of the local solver, the method may become inefficient for large problems. For example, in the healthcare context, when the service times for patients are relatively short, each caregiver can visit more patients. In this case, optimizing the tasks assignment and scheduling the tasks for a pair of caregivers in a local problem may become difficult. In the paper, we improve the gossip by using a CG-based solver for the local problems

and show that the resulting algorithm (gossip-CG) outperforms the standard CG for large problems, and for some problem instances is superior to gossip-CPLEX.

A glimpse into the parallelization results As mentioned earlier, in Paper 4, we proposed a parallel version of the gossip algorithm. In general, we showed for several problem instances that Algorithm 4 converges faster, compared to Algorithm 3. As an example, this can be seen in Figure 4.7, where the convergence behavior of Algorithm 4 for different numbers of cores is depicted for a problem instance called r109. It is seen that both gossip with 2-threads and 4-threads outperforms the single-threaded version, which corresponds to Algorithm 3. Although in the beginning the two-threaded version gives better improvement of the cost function due to randomness of the search moves, by the end of the time limit of 1800 s the 4-threaded version manages to find a better solution.

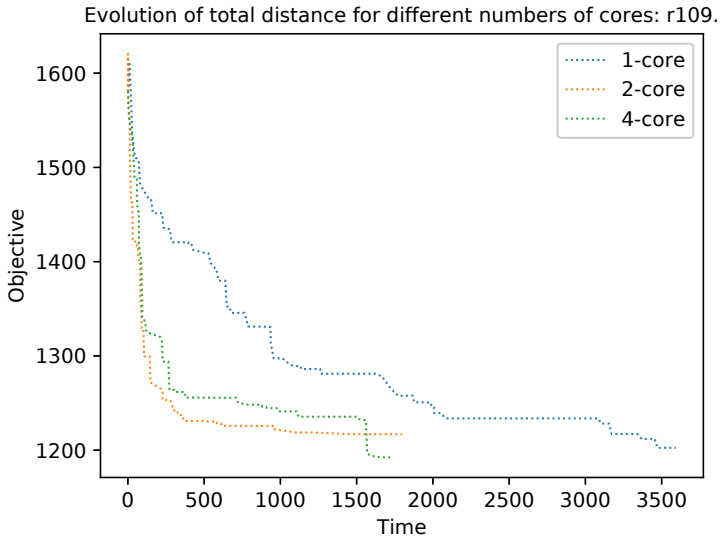

Figure 4.7: Evolution of the objective function for a small instance of VRPTW.

4.6 Summary

In this chapter, we provided the reader with some background on vehicle routing problems decompositional algorithms used in the thesis. Specifically, we formalized a Benders decomposition for HMVRPs. Also, using a small example, we demonstrated how tasks are assigned to robots in the master problem, and how they are scheduled in the subproblmes. Column generation, as another decompositional algorithm used in the thesis was also explained, together with its relationship with Benders decomposition.

Additionally, the framework of the gossip algorithm was explained with an illustrative example. We then presented the architecture of the gossip-CG algorithm, which was used in Paper 3. We also mentioned our case study, i.e. the home healthcare routing and scheduling problem, on which we tested our algorithms.

Finally, we presented the parallelized version of the gossip algorithm, which was published in Paper 4. A sample of the performance improvements gained by this algorithm was also given.

Answering the research questions In Section 1.3, we formulated two research questions regarding moving devices, which are revisited, as follows.

RQ3: In the context of routing of moving devices, how can a known efficient algorithm be integrated with a metaheuristic to solve larger problems?

RQ4: How can the performance of the integrated optimization approach developed in response to *RQ3* be improved by better utilization of existing hardware?

To answer *RQ3*, in Paper 3, we proposed the gossip-CG algorithm, which was an integration of the gossip and column generation algorithms. It proved to outperform CG on the many problem instances we used to test it. Gossip-CG consistently yielded solutions with higher quality compared to those obtained from pure CG. Moreover, gossip-CG was able to rapidly improve the initial solution even after only a few iterations.

To answer *RQ4*, in Paper 4, we utilized the distributed nature of the gossip algorithm and proposed its multi-threaded version, which better utilizes a multi-core CPU. The implementation effort was also proved to be minimal. Hence, our answer to *RQ4* is to use algorithms that can be easily parallelized.

Chapter 5

Energy Efficient Routing and Scheduling of AGVs

This chapter gives an overview of our work in Paper 5 and Paper 6. Following our work on energy optimization of industrial robots and vehicle routing problems, we pursued another research topic that could benefit from those results. Energy efficient routing and scheduling of AGVs was then chosen, as it seemed like an interesting bridge between robot energy optimization, and routing/scheduling problems. Our work started in the spring of 2018 together with a Swedish AGV manufacture called AGV Electronics (AGVE), which is now owned by a Japanese company called Murata Machinery. Apart from the interesting research challenges in the area, as mentioned in Chapter 1, there was an extra incentive to work on this problem due to existing industrial needs. In fact, there is a growing demand for more AGV systems in both manufacturing and non-manufacturing environments, as well as the pressure for more sustainable solutions.

To give a better picture of the competition and demand for AGVs, consider the following statistics. Currently, there are more than 100 AGV manufacturers in the world. In terms of sold units, almost $111,000$ logistic systems were sold in 2018, which compared to $69,000$ units sold in 2017 means that the market has increased almost 60% in one year. Of those sold units, $7,700$ were AGVs in manufacturing environments and almost 103,000 in non-manufacturing environments [74]. There has been a trend in major e-commerce companies to use AGV solutions, which drives this growing market. Furthermore, another sector that could benefit from AGVs is the healthcare sector, which could use them in hospitals. Furthermore, the global automated guided vehicle market size was valued at USD 3.0 billion in 2019 and is expected to witness a compound annual growth rate (CAGR) of 14.1% from 2020 to 2027 [75]. Of course this figure does not consider the looming global recession due to the ongoing epidemic Coronavirus.

5.1 Challenges and opportunities

There are several challenges and opportunities in the field of AGV routing and scheduling. For example, although AGV manufacturers have developed rather

efficient control policies and algorithms, retrofitting the existing heuristic to future's more dense, more complicated, and more demanding AGV layouts, is not guaranteed to be easy. Currently, it is common to use heuristics to allocate vehicles to orders and route them. There are also rules of thumbs to avoid collisions and deadlocks, which are not guaranteed to work all the time, and might even hinder the performance.

It is also noteworthy that, although AGVs are primarily used to increase productivity, sometimes the installed systems cannot meet the required performance metrics, such as throughput, which is set by the AGV customers. To compensate for that, often the customers use manual forklifts in conjunction with the AGVs. This underperformance can be improved, for example, by optimizing task assignments, routing, and scheduling. With increasing demand for high-performance AGV solutions, it is appealing to employ optimization algorithms that handle the order allocation, scheduling, routing, and deadlock avoidance in a more efficient way.

Furthermore, energy optimization of AGVs is not a well-explored research field, and it is not the first priority of AGV manufacturers who often struggle with delivering the required output. However, there are several benefits to reducing energy consumption of AGVs. Apart from the obvious benefit of reduced energy costs, energy efficient routing and scheduling cuts down on the frequency of battery charging. This reduces the down time of vehicles. Additionally, higher energy efficiency could translate into the smaller required battery capacity, which in turn, leads to cheaper and more competitive products. In this thesis, we aim to present an improved method to tackle the above-mentioned issues, with promising results.

5.2 The approach

As mentioned in Chapter 1, the common control strategies for AGV systems can be divided into two groups: centralized methods and decentralized ones. In centralized methods, a single controller tackles task assignment and routing of the vehicles, as in [26], whereas in a decentralized strategy these roles are partly or entirely delegated to the vehicles. For example, the authors in [27] propose a decentralized method where first the vehicles assign tasks to themselves such that a global objective function is minimized, namely the total completion time. Then the vehicles move towards their destinations using a decentralized algorithm.

In this thesis, we limit our scope to a centralized control approach that handles tasks assignments and routing. The reason for this choice is twofold. Firstly, the AGV company that we have worked with utilizes such a control strategy. Hence, to compare our potential solutions with their technology, and to be able to effectively utilize their resources, tools, and expertise it was imperative to work on a similar control strategy. Secondly, it has always been the author's priority to develop methods that can be integrated into existing hardware with minimum intrusion and modifications. This would maximize the chance for the developed solution to be adopted by the company. Note that, to develop a decentralized method, one major assumption is that the vehicles should be able to communicate with each other and be somewhat *intelligent*.

For example, vehicles should know where they are in the layout. However, in the current technology developed by the company, the vehicles communicate only with the central traffic controller, and are unaware of other vehicles and even their own location in the system. Thus, switching to a decentralized approach, regardless of its potential benefits, would require major hardware-related changes in the vehicles, and the way communication has been implemented. This would diminish the desire for such a control policy by the company. Therefore, we decided to focus on a centralized approach.

Moreover, as mentioned in Chapter 1, common optimization methodologies for scheduling and routing of AGVs can be categorized into exact and heuristic mathematical methods, simulation studies, metaheuristic techniques and artificial intelligence (AI) based approaches, as surveyed in [25]. Due to our interest and background in mathematical optimization and constraint programming, in development of our methodologies we have been mainly concerned with such methods.

5.3 Case study: Volvo

Our case study is based on an FMS belonging to Sweden's largest automobile manufacturer, Volvo Cars. The layout of the FMS spans over an area of roughly 85000 m^2, which consists of several robot cells and conveyors. AGVs are employed to transport racks, loaded with raw sheet metal parts, from inbound conveyors to robot cells for various processes, and then to ship the empty racks back to outbound conveyors. The current AGV solution has been deployed by AGVE. A movie clip of the AGVs in the plant can be accessed from https://bit.ly/2KuTaQu. Furthermore, Figure 5.1 shows loaded and empty vehicles at the factory floor.

Figure 5.1: AGVs in the Volvo Cars plant. The left vehicle carries a rack loaded with sheet metal parts.

The layout The mixed graph of the transport system includes 334 nodes and 412 arcs, out of which 98 are undirected. There are 21 home locations and several charging stations scattered around the entire layout. The current system consists of 19 AGVs moving on a layout whose graph is depicted in Figure 4.1. In the figure, a few of the robot stations are annotated with their names, for example *ST8110*. Most

of the inbound and outbound conveyors are concentrated on the right side of the figure. They are prefixed with *In* and *Out* respectively.

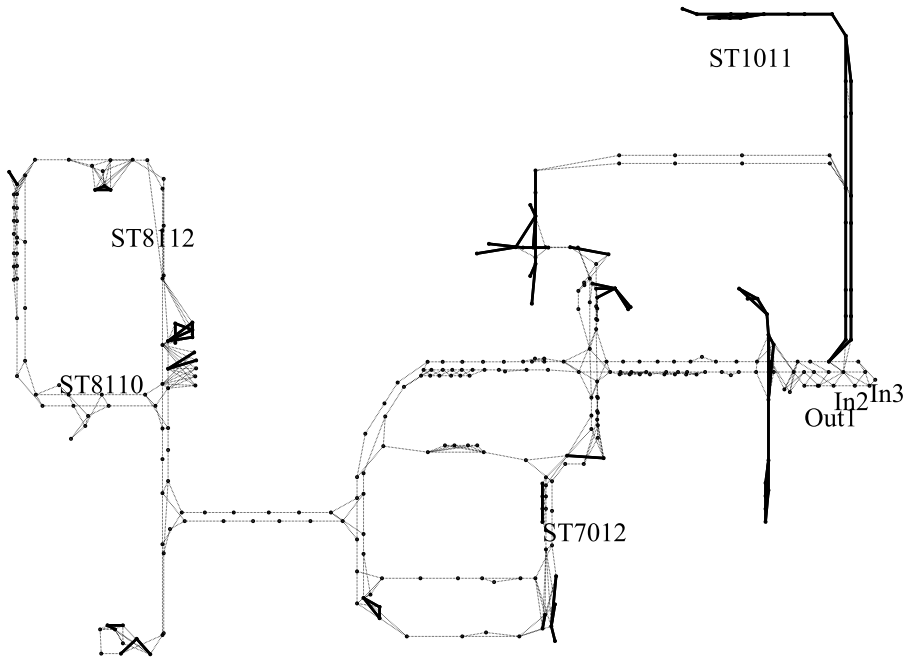

Figure 5.2: Graph of the transportation system with 334 nodes and 412 arcs. Four robot stations and three conveyors are annotated, prefixed with *ST* for robot stations, with *In* for the inbound conveyors, and with *Out* for the outbound one. The thin arcs are directed while the thick ones are undirected.

Traffic controller Since the existing traffic controller designed by AGVE is centralized, it operates based on a few simple principles. (1) a shortest path algorithm like Dijkstra [76] is used to generate each vehicle's routes. (2) a vehicle is assigned to a task only if it is within a prescribed distance to the node corresponding to the pick subtask, which could either be a station or a conveyor. Therefore, an order could be waiting in the queue while there is an unassigned vehicle in a remote location of the layout. (3) If a task is tardy enough, even a far-off AGV can be assigned to it if no other vehicle is in the vicinity. (4) The traffic controller routinely updates the task assignments and routes with the latest changes/disruptions. (5) a so-called *order stealing* may occur; a vehicle that has just finished its assignment may steal an order from another vehicle, if the latter is far away.

The optimization model and algorithm The optimization model, is based on a Benders decomposition method proposed in [26]. For more information on Benders decomposition, see Section 4.2. In that article, the authors employed a variation of Benders decomposition to solve the collision-free AGV scheduling and routing problem in two stages for problems up to 6 AGVs and 13 tasks in a graph of 30 nodes. Their approach tackled the task allocation and scheduling using a CP

model with tardiness as its cost function, and the routing problem was handled using a Mixed MILP, which was basically a CSP. Their proposed CSP model, however, has difficulty scaling to large systems. Furthermore, the method is limited to AGV layouts with special design, namely those with equidistant nodes on the tracks. As our first contribution to the field, in Paper 5, we introduce several extensions and improvements to [26] to allow solving larger problems, with promising results.

Reducing energy consumption In one of the few publications addressing the energy optimization of AGVs [32], the authors formulate the problem as VRP, combined with an energy consumption model of the AGV that takes into account the load. However, results of such analysis cannot be readily used to control the vehicles, mainly due to lack of regards for time and collision between vehicles. As our next contribution to the field, in Paper 6, we show that by implementing an effective scheduling and routing algorithm developed in Paper 5, it is possible to improve system efficiency so much that it allows for reduction of speed to save energy while still outperforming the original solution. An important advantage of such an energy reduction method is simplicity of its implementation in the real system, since regulating the vehicle's speed can be done conveniently. The method leads to significant improvements in key performance measures such as makespan, as well as energy optimization. The reader is referred to the two papers for full detail of the optimization model, experiments, and results.

A glimpse into the results Now we give a glimpse to one of our findings. From the experiments we observe that our scheduling and routing algorithm improves the makespan of the system significantly, compared the original traffic controller. Now, to save some energy, the speed limit of AGVs is reduced. It is then observed that the improvements in makespan deteriorates, which is expected. Yet, even after the speed reduction, the makespan remains better than the original solution. This is further illustrated in Figure 5.3.

This trend continues until the max speed of 0.8 m/s is reached, where the sum of tardiness is no longer better than that of the original solution. The important result is that until reaching that speed, reducing the maximum speed implies that the energy consumption is decreased up to 34%, while outperforming the original solution in all performance measures.

The important finding is that, for a given makespan (1830 for the original solution, which amounts to 100% in the figure), our method can reduce the energy consumption by around 38%, while the makespan remains better than the original solution from the traffic controller. The data points for the original solution are the overlapping points in the upper right corner of the figure.

The optimization results clearly show that an effective scheduling and routing algorithm can be combined with reduction of the top speed of the AGVs as an effective strategy to improve multiple performance measures such as makespan, lateness, tardiness, and energy.

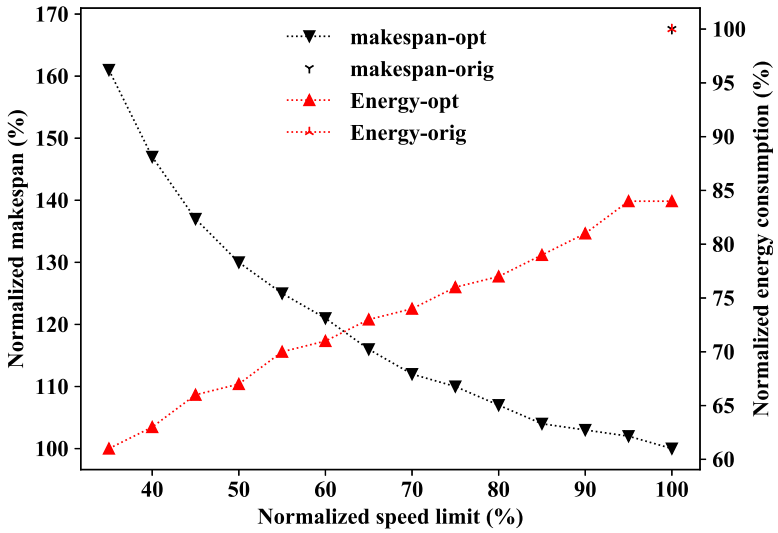

Figure 5.3: Makespan and total energy consumption vs. speed limit for 10 tasks and four AGVs.

5.4 Final remarks

Now we make a few final remarks, namely on deadlocks, an additional idea that we explored in the hope of improving the results, and the foreseen future of our AGV research.

A note on deadlocks A deadlock is a situation involving opposite parties, where no progress can be achieved due to a fundamental disagreement. In other words, a deadlock occurs when a process is waiting for an event or action that will never occur [77]. An event can be, for example, releasing a booked resource, such as a physical location, a file, etc.

Consider two processes, P1 and P2 and two resources R1 and R, where P1 and P2 have booked R1 and R2 respectively. Furthermore, P1 is waiting for P2 to release R1 before it can be completed, and P2 is likewise waiting for P1 to release R1. Clearly neither of the resources will be released and the situation will not resolve without external intervention. An effective tool to analyze deadlocks is a *resource allocation* graph, which is a directed bipartite graph and can be used to design deadlock avoidance algorithms, as in [78]. When a resource is booked by a process, an arrow emanates from the resource to the process. When a process requests a resource, the arrow is from the process towards the resource. This is depicted in Figure 5.4, where the deadlock situation of the example above is seen.

In the context of AGVs resources can be physical locations, which can be modeled by nodes and arcs in a graph representation of an AGV layout. Deadlocks may occur, for instance, when two AGVs that have booked two nodes or arcs request from each other the release of a booked resource so they can move. Now, we describe a deadlock situation in an AGV system.

In Figure 5.5 a partial AGV layout with two forklift AGVs are depicted. In the layout, there are four lanes that are two by two parallel and in opposite directions.

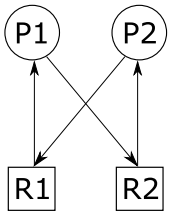

Figure 5.4: A deadlock situation shown using resource allocation graph

Hence, the travel in each direction is done in a unique lane. AGV 1 has booked nodes A and B, while AGV 2 has booked nodes C and D. However, due to physical proximity of the curves near the junction and the size of the vehicles, when AGV 1 wants to travel on the arc AB, it must also book the arc CD to avoid possible collision. Similarly, when the second AGV wants to travel on the curve CD it must also book the arc AB. In the present situation both vehicles are waiting for each other to release the required resource and are deadlocked. Deadlock situations in AGV systems can be avoided through several methods.

Deadlocks of this type maybe avoided by *zone control* in real time. This technique subdivides the guide paths into disjoint zones that represent intersections of several paths (like the example above), workstations, etc. Then, the access to each zone is authorized by a control policy in advance [79] to avoid deadlock and collision. It is also possible to schedule the movements of the vehicles in advance to avoid deadlocks and collisions, which has been addressed in [26] and is also the method of choice in the present work.

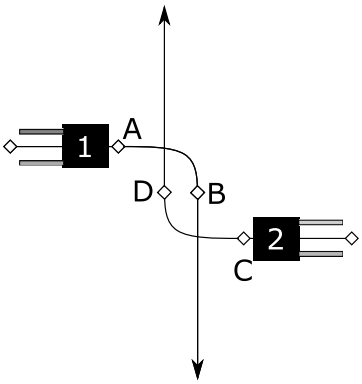

Figure 5.5: AGV 1 and AGV2 are deadlocked.

Exploring the idea of abstraction In the field of discrete-event systems, one common problem is the *state-space explosion*, which means that the size of the state-space system grows exponentially with the number of local state variables. The problem can be coped with using techniques such as abstraction [80], [81]. The idea is to examine the graph (automaton) describing the system, and merge nodes (states) that are not of immediate importance, hence drastically reducing the required memory and computational effort. Thus, one idea we experimented with was reducing the size of the layout's graph to improve the computational speed.

In the AGV layout shown in Figure 5.2, we can see long pathways that consist of nodes with only one input and one output arc. One can combine these nodes such that a long stretch is replaced by one arc (with adjusted weight) that ends in one input node with two or more input arcs and one output node with two or more output arcs, as shown in Figure 5.6.

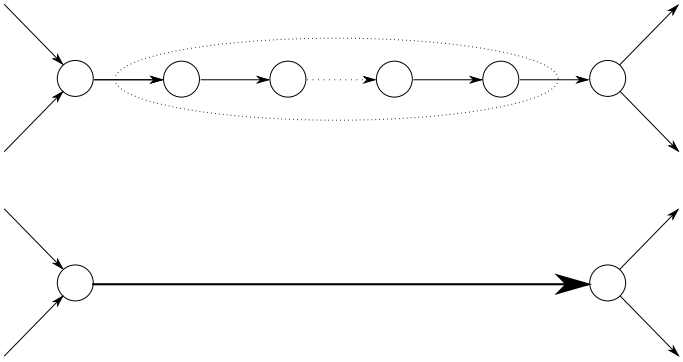

Figure 5.6: Abstraction; merging nodes with one input and one output arc.

We developed an algorithm to achieve this and applying it to the graph reduced the number of nodes from 334 to 296, or about 11%. For example, for a case with 10 tasks and 4 AGVs and algorithm `heur-Benders-CP`, the CPU time reduces from 34.07 s to 24.20 s. It should also be pointed out that in a long unidirectional path with multiple nodes, more than one AGV can travel simultaneously. Therefore, when the logical nodes are merged, the capacity of the abstracted nodes should also be adjusted appropriately to reflect the physical capacity; otherwise the makespan will deteriorate. For example, if the capacity adjustments are not made, in the same problem instance, the makespan increases from 1064 s to 1090 s.

5.5 Summary

In this chapter, we provided an overview of the problem of energy efficient routing and scheduling of AGVs. The challenges and opportunities were reviewed, as well as our methodology for dealing with the problem. We showcased some of our results from the papers.

Answering the research questions In Section 1.3, we formulated two research question regarding AGVs, which are revisited, as follows.

RQ5: How can conflict-free routing and scheduling of moving devices be addressed using general-purpose solvers for large-scale problems?

RQ6: How can the energy saving methodology developed in response to *RQ1-RQ2* be reused in the context of large-scale routing and scheduling of moving devices?

To answer *RQ5*, in Paper 5, we proposed a methodology based on Benders decomposition, which utilized the power of CP and SMT.

To answer *RQ6*, in Paper 6, we first showed that the important contributing factors in energy consumption of AGVs were traveled distance and velocity, while acceleration was not as significant as it was for industrial robots. Then, we showed that in conjunction with the methodology developed above, the vehicles could be slowed down to save energy, while the performance indexes remained better than those of the original traffic controller.

Chapter 6

Summary of Included Papers

In this chapter, we provide a brief summary of the articles included in the thesis. All of the papers can be found in Part II, in the original format of their corresponding conference or journal.

Paper 1

Sarmad Riazi, Kristofer Bengtsson, Oskar Wigström, Emma Vidarsson, and Bengt Lennartson. *Energy optimization of multi-robot systems.* Proceedings of the 11th IEEE Conference on Automation Science and Engineering (CASE), Gothenburg, 2015, pp. 1345–1350.

Paper 1 introduces a novel optimization procedure to reduce energy consumption of industrial robots in a multi-robot environment. The procedure includes four steps; (1) programming/running an arbitrary motion on the robot, (2) logging the motion via existing software functionalities of the robots (3) optimizing the motion offline, and (4) running the optimized motion on the robot. The optimization model can be solved quickly with an open-source nonlinear programming solver. Two important features of the optimization procedure are that paths are preserved, and the original cycle time can be retained. A few scenarios have been studied using a real industrial robot.

Paper 2

Sarmad Riazi, Oskar Wigström, Kristofer Bengtsson, and Bengt Lennartson. *Energy and Peak-power Optimization of Time-bounded Robot Trajectories.* IEEE Transactions on Automation Science and Engineering, 2017, 14(2): 646–657.

This paper builds upon the optimization procedure presented in Paper 1 with several extensions. Firstly, various case studies are carried out using robots of different sizes. Multi-robot scenarios are studied in more detail using a real four-robot station. Furthermore, different surrogate objective functions are employed to target the energy consumption and peak-power of the robots. It is also shown that the optimization technique yields results that are superior to the existing

energy reduction measures built-in to the robots. One extra feature of this study is the careful energy measurements at a relatively constant temperature during the experiments. This is to minimize the effect of temperature on energy consumption and thereby obtaining more reliable results.

Paper 3

Sarmad Riazi, Oskar Wigström, Kristofer Bengtsson, and Bengt Lennartson. *A Column Generation-based Gossip Algorithm for Home Healthcare Routing and Scheduling Problems.* IEEE Transactions on Automation Science and Engineering, 2019, 16(1):127–137.

Paper 3 concerns a integrated/hybrid optimization approach to solve the home healthcare routing and scheduling problem as a vehicle routing problem. The method breaks the problem into smaller local problems within the framework of a distributed algorithm called gossip, and solves the local problem using the column generation algorithm. It is shown that the integration of the gossip algorithm with column generation outperforms the standard column generation.

Paper 4

Sarmad Riazi, Oskar Wigström, Kristofer Bengtsson, and Bengt Lennartson. *Parallelization of a gossip algorithm for vehicle routing problems.* Proceedings of the 14th IEEE Conference on Automation Science and Engineering (CASE), Munich, 2018, pp. 92–97.

The gossip algorithm, which was applied to the routing problems in Paper 3, naturally decomposes the problem in local problems that can be solved in parallel. This paper contributes to the field by proposing a parallelized gossip algorithm that takes advantage of multi-core processors to speed up the solution process. Benchmark instances are solved considerably quicker using the proposed method.

Paper 5

Sarmad Riazi, Thomas Diding, Petter Falkman, Kristofer Bengtsson, and Bengt Lennartson. *Scheduling and Routing of AGVs for Large-scale Flexible Manufacturing Systems.* Proceedings of the 15th IEEE Conference on Automation Science and Engineering (CASE), Vancouver, 2019, pp. 891–896.

In Paper 5, we propose a heuristic to solve the problem of scheduling and routing of automated guided vehicles in a large-scale manufacturing system, with promising results. The algorithm is based on an existing integrated method [26], which we improve such that it becomes capable of handling much larger systems. Additionally, we study the use of a SAT solver as the main optimization engine, which proves to be a viable option in absence of commercial CP solvers. Our case study concerns a real large-scale AGV system installed at Volvo Cars.

Paper 6

Sarmad Riazi, Kristofer Bengtsson, and Bengt Lennartson. *Energy Optimization of Large-scale AGV Systems.* Conditionally accepted for publication in IEEE Transactions on Automation Science and Engineering.

Our final paper serves as a bridge between Paper 1-2, which dealt with energy optimization of robots, and Paper 3-4, which addressed routing and scheduling of vehicle/caregivers with focus on performance. Paper 6 is based on Paper 5, and has several contributions; (1) Extension of the mathematical model in Paper 5 to take into account additional constraints in the FMS at Volvo Cars, (2) low-level energy consumption analysis of the AGVs, which is used to design an energy saving strategy, (3) Comparison between the outcome of our scheduling/routing method with the existing traffic controller used in the AGV system, with promising results in terms of several performance measures, and (4) detailed analysis of the proposed energy saving strategy using data from the real system.

Chapter 7

Concluding Remarks

This thesis introduced several methods for energy and route optimization of moving devices with promising results. We have dealt with energy optimization of robots, vehicle routing problems, and energy efficient routing and scheduling of AGVs, which combine the former two problems. As the conclusions are grouped around the research questions (RQ), we start by the first one.

RQ1: *How can robot's energy consumption be reduced in a non-intrusive way, while preserving original path and cycle time?*

In Paper 1, we proposed a methodology for energy optimization of robots that preserve the path and cycle time using the built-in robot functionalities. This means that no additional change is needed to be made in the robot controller, which is one measure of being non-intrusive. Moreover, since our methodology only changes trajectories but not paths, it reduces the concern for possible collisions. This is another measure of being non-intrusive.

Another feature of the technique is that it does not require confidential robot parameters, which makes the method even more appealing. An important advantage of the methodology is its simplicity. The author heard from a professor at Chalmers University that the best ideas are often the simple ones.

In fact, out methodology is so powerful and straightforward that, in collaboration with University West in Sweden, it has also been implemented in a real-world case study that considers the multi-robot material handling system of a multi-stage tandem press line, also with promising results in reduction of energy consumption [82].

RQ2: *In what ways can energy and peak-power for multi-robot cell be reduced?*

The short answer is, in addition to the technique developed in Paper 1, one should slow down the robot motions when it is possible. For example, when one robot is supposed to wait for another one, it can be programmed in a way that it moves slower. This way, the cycle time of the robot cell will not be affected.

To achieve this, in Paper 2 we refined our procedures further. We employed accurate energy measurement equipment and provided procedures to have

reliable measurements and replicable experiments. A detailed case study and its results are presented in Paper 2. The study was conducted at Daimler. In short, we showed that it was possible to reduce up to 30% of energy consumption and up to 60% of peak-power.

RQ3: *In the context of routing of moving devices, how can a known efficient algorithm be integrated with a metaheuristic to solve larger problems?*

The author's answer is that, one way is to integrate the efficient algorithm as a local solver into a suitable metahuristic framework. For example, this can be done by decomposing the problem into local problems. This may lead into an algorithm that is more efficient than the original centralized algorithm.

To implement this idea, in Paper 3, we proposed the gossip-CG algorithm, which was an integration of the gossip and column generation algorithms. It proved to outperform CG on the many problem instances we used to test on. Gossip-CG consistently yielded solutions with higher quality compared to those obtained from pure CG. Moreover, gossip-CG was able to rapidly improve the initial solution even after only a few iterations.

RQ4: *How can the performance of the integrated optimization approach developed in response to RQ3 be improved by better utilization of existing hardware?*

Our experience is that, one approach could be parallelization of the integrated algorithm, or in general, any other optimization algorithm. But parallelization is not always an option, as not all algorithms is easily adopted for this. Furthermore, the implementation effort can be significant. Thus, a metaheuristic that allows for parallelization can be a good candidate.

To validate the idea, in Paper 4 we utilized the distributed nature of the gossip algorithm and proposed its multi-threaded version, which better utilizes a multi-core CPU. The implementation effort was also proved to be minimal. Hence, our answer to *RQ4* is to use algorithms that can be easily parallelized.

RQ5: *How can conflict-free routing and scheduling of moving devices be addressed using general-purpose solvers for large-scale problems?*

To tackle this question, we focused on an AGV routing and scheduling problem, as a particular case of moving devices. Our approach has been to generate schedules that are collision and deadlock-free, rather than a scheduling method that also has to handle deadlocks on the fly, if they occur. Moreover, according to our experience, much improvement can be done by better assignments of tasks to AGVs, in addition to better scheduling.

Since our problem, apart from the deadlock-free routing part, was similar to a job-shop scheduling problem, we thought about using general-purpose CP or SMT solvers, that are known to be efficient for this kind of problems [83]. We also had the intuition that a decompositional algorithm could be suitable. Indeed, both of the ideas were already published [26]. However, that article's approach was not flexible enough to handle the AGV layout that we were

dealing with. Besides, it was not suitable for very large layouts, which was our case.

Therefore, in Paper 5 we proposed a methodology based on [26], which utilized the power of CP and SMT in a decomposition framework. We modeled the problem in a smarter way, which eliminated the shortcomings in [26]. Indeed, we also proposed a heuristic version of the algorithm, with higher performance.

RQ6: *How can the energy saving methodology developed in response to RQ1-RQ2 be reused in the context of large-scale routing and scheduling of moving devices?*

To answer this question in Paper 6, we first showed that the important contributing factors in energy consumption of AGVs were traveled distance and velocity, while acceleration was not as significant as it was for industrial robots. Then, we showed that in conjunction with the methodology developed above, the vehicles could be slowed down to save energy, while the performance indexes remained better than those of the original traffic controller.

Future work At the time of writing this thesis, the author has started working at AGVE as a research engineer. The author is proud to announce that his current job description includes developing routing and scheduling algorithms for the next generation of traffic controllers. The outcome should be delivered within a five-year plan, in collaboration with Murata Machinery. The ongoing plans for the immediate future are twofold. First, the author is exploring online deadlock and blocking avoidance techniques, using information from the generated route and nodes that are to be visited soon. For example, if two vehicles are predicted to end up in a tight area with risk of getting deadlocked, one will be slowed down, stopped, or re-routed. The second task is to expand and further improve the algorithms and methods developed in Paper 5 and Paper 6.

Another idea that will also be explored, is the use of AI for scheduling the AGVs even better. For example, consider an FMS with two work shifts. In the first work shift, some areas of the system are more active, while in the second work shift, other areas are generating transport orders more frequently. When an AGV has delivered its order, it can be sent to areas where it is expected to see more request, rather than simply sending the AGV into the nearest home location. Therefore, the activity pattern of the system can be *learned* and used as an input into the control system.

Bibliography

[1] M. Pellicciari, A. Avotins, K. Bengtsson, G. Berselli, N. Bey, B. Lennartson, and D. Meike, "Areus - innovative hardware and software for sustainable industrial robotics," in *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*, Aug. 2015, pp. 1325–1332 (cit. on pp. 4, 27).

[2] A. Kobetski and M. Fabian, "Velocity balancing in flexible manufacturing systems," in *2008 9th International Workshop on Discrete Event Systems*, May 2008, pp. 358–363 (cit. on pp. 4, 27).

[3] A. Vergnano, C. Thorstensson, B. Lennartson, P. Falkman, M. Pellicciari, F. Leali, and S. Biller, "Modeling and optimization of energy consumption in cooperative multi-robot systems," *Automation Science and Engineering, IEEE Transactions on*, vol. 9, no. 2, pp. 423–428, Apr. 2012, ISSN: 1545-5955. DOI: 10.1109/TASE.2011.2182509 (cit. on pp. 4, 27).

[4] O. Wigström, B. Lennartson, A. Vergnano, and C. Breitholtz, "High-level scheduling of energy optimal trajectories," *Automation Science and Engineering, IEEE Transactions on*, vol. 10, no. 1, pp. 57–64, Jan. 2013 (cit. on pp. 4, 27).

[5] M. Todtermuschke, M. Findeisen, and A. Bauer, "Methodology for creation a reference trajectory for energetic comparability of industrial robots in body shop," *Procedia CIRP*, vol. 23, pp. 122–126, 2014 (cit. on pp. 4, 27).

[6] N. Wemhöner, "Flexibilitätsoptimierung zur Auslastungssteigerung im Automobilrohbau," PhD thesis, Aachen, 2006, XI, 207 S. : Ill., graph. Darst. (Cit. on pp. 4, 27).

[7] D. Meike and L. Ribickis, "Energy efficient use of robotics in the automobile industry," in *Advanced Robotics (ICAR), 2011 15th International Conference on*, Jun. 2011, pp. 507–511 (cit. on pp. 4, 27).

[8] J. Engelmann, *Methoden und Werkzeuge zur Planung und Gestaltung energieeffizienter Fabriken*. IBF, 2009 (cit. on pp. 4, 27).

[9] S. Riazi, K. Bengtsson, O. Wigström, E. Vidarsson, and B. Lennartson, "Energy optimization of multi-robot systems," in *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*, Aug. 2015, pp. 1345–1350. DOI: 10.1109/CoASE.2015.7294285 (cit. on pp. 4, 28).

[10] S. Riazi, O. Wigström, K. Bengtsson, and B. Lennartson, "Energy and peak power optimization of time-bounded robot trajectories," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 646–657, Apr. 2017 (cit. on pp. 4, 28).

[11] S. Riazi, C. Seatzu, O. Wigström, and B. Lennartson, "Benders/gossip methods for heterogeneous multi-vehicle routing problems," in *Emerging Technologies Factory Automation (ETFA), 2013 IEEE 18th Conference on*, 2013, pp. 1–6. DOI: 10.1109/ETFA.2013.6647983 (cit. on p. 5).

[12] N. Genet, W. Boerma, M. Kroneman, A. Hutchinson, and R. Saltman, "Home care across europe," *Observatory Studies Series*, p. 156, 2013 (cit. on p. 5).

[13] E. Cheng and J. L. Rich, "A home health care routing and scheduling problem," Tech. Rep., 1998 (cit. on p. 5).

[14] M. Cissé, S. Yalçındağ, Y. Kergosien, E. Şahin, C. Lenté, and A. Matta, "Or problems related to home health care: A review of relevant routing and scheduling problems," *Operations Research for Health Care*, vol. 13-14, pp. 1–22, 2017 (cit. on p. 5).

[15] B. Kallehauge, "Formulations and exact algorithms for the vehicle routing problem with time windows," *Computers & Operations Research*, vol. 35, no. 7, pp. 2307–2330, 2008. DOI: http://dx.doi.org/10.1016/j.cor.2006.11.006 (cit. on p. 5).

[16] R. Baldacci, A. Mingozzi, and R. Roberti, "Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints," *European Journal of Operational Research*, vol. 218, no. 1, pp. 1–6, 2012. DOI: http://dx.doi.org/10.1016/j.ejor.2011.07.037 (cit. on p. 5).

[17] ——, "New route relaxation and pricing strategies for the vehicle routing problem," *Operations Research*, vol. 59, no. 5, pp. 1269–1283, 2011. DOI: 10.1287/opre.1110.0975 (cit. on p. 5).

[18] M. Desrochers, J. Desrosiers, and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Operations Research*, vol. 40, no. 2, pp. 342–354, 1992 (cit. on p. 5).

[19] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "Heuristics for multi-attribute vehicle routing problems: A survey and synthesis," *European Journal of Operational Research*, vol. 231, no. 1, pp. 1–21, 2013, ISSN: 0377-2217 (cit. on p. 5).

[20] S. Riazi, P. Chehrazi, O. Wigström, K. Bengtsson, and B. Lennartson, "A gossip algorithm for home healthcare scheduling and routing problems," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 10 754–10 759, 2014, 19th IFAC World Congress (cit. on p. 5).

[21] S. Riazi, O. Wigström, K. Bengtsson, and B. Lennartson, "Decomposition and distributed algorithms for home healthcare routing and scheduling problem," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2017, pp. 1–7 (cit. on p. 5).

[22] S. Riazi, O. Wigström, K. Bengtsson, and B. Lennartson, "A column generation-based gossip algorithm for home healthcare routing and scheduling problems," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 1, pp. 127–137, Jan. 2019 (cit. on p. 6).

[23] S. Riazi, K. Bengtsson, and B. Lennartson, "Parallelization of a gossip algorithm for vehicle routing problems," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, Aug. 2018, pp. 92–97 (cit. on p. 6).

[24] K. C. T. Vivaldini, L. F. Rocha, M. Becker, and A. P. Moreira, "Comprehensive review of the dispatching, scheduling and routing of agvs," in *CONTROLO'2014 – Proceedings of the 11th Portuguese Conference on Automatic Control*, A. P. Moreira, A. Matos, and G. Veiga, Eds., Cham: Springer International Publishing, 2015, pp. 505–514 (cit. on p. 6).

[25] H. Fazlollahtabar and M. Saidi-Mehrabad, "Methodologies to optimize automated guided vehicle scheduling and routing problems: A review study," *Journal of Intelligent & Robotic Systems*, vol. 77, no. 3, pp. 525–545, Mar. 2015 (cit. on pp. 6, 61).

[26] A. I. Corréa, A. Langevin, and L. M. Rousseau, "Scheduling and routing of automated guided vehicles: A hybrid approach," *Computers & Operations Research*, vol. 34, no. 6, pp. 1688–1707, 2007, ISSN: 0305-0548 (cit. on pp. 6, 7, 60, 62, 63, 65, 70, 74, 75).

[27] M. P. Fanti, A. M. Mangini, G. Pedroncelli, and W. Ukovich, "A decentralized control strategy for the coordination of AGV systems," *Control Engineering Practice*, vol. 70, pp. 86–97, 2018 (cit. on pp. 6, 60).

[28] J. N. Hooker, "Planning and scheduling by logic-based Benders decomposition," *Operations Research*, vol. 55, no. 3, pp. 588–602, 2007 (cit. on p. 7).

[29] S. Riazi, T. Diding, P. Falkman, K. Bengtsson, and B. Lennartson, "Scheduling and routing of AGVs for large-scale flexible manufacturing systems," in *Automation Science and Engineering (CASE), 2019 IEEE International Conference on*, to appear, Aug. 2019 (cit. on p. 7).

[30] S. Liu and D. Sun, "Minimizing energy consumption of wheeled mobile robots via optimal motion planning," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 2, pp. 401–411, Apr. 2014 (cit. on p. 7).

[31] Yongguo Mei, Yung-Hsiang Lu, Y. C. Hu, and C. S. G. Lee, "Deployment of mobile robots with energy and timing constraints," *IEEE Transactions on Robotics*, vol. 22, no. 3, pp. 507–522, Jun. 2006 (cit. on p. 7).

[32] L. Qiu, J. Wang, W. Chen, and H. Wang, "Heterogeneous agv routing problem considering energy consumption," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec. 2015, pp. 1894–1899 (cit. on pp. 7, 63).

[33] S. Riazi, K. Bengtsson, and B. Lennartson, "Energy optimization of large-scale AGV systems," *IEEE Transactions on Automation Science and Engineering*, pp. 1–12, Jan. 2020. DOI: 10.1109/TASE.2019.2963285 (cit. on p. 7).

[34] J. Hooker, *Integrated methods for optimization*. New York, NY: Springer, 2010 (cit. on pp. 11, 23, 24, 45, 46).

[35] H. Williams, *Model Building in Mathematical Programming*. Wiley, 2013 (cit. on p. 13).

[36] I. Lustig and J.-F. Puget, "Program does not equal program: Constraint programming and its relationship to mathematical programming," *Interfaces*, vol. 31, pp. 29–53, Dec. 2001 (cit. on pp. 13, 18, 19).

[37] D. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *Journal of the ACM*, vol. 51, Nov. 2001 (cit. on p. 14).

[38] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations Research*, vol. 8, no. 1, pp. 101–111, 1960. DOI: 10.1287/opre.8.1.101 (cit. on pp. 14, 51).

[39] J. Desrosiers and M. E. Lübbecke, "A primer in column generation," in *Column generation*, G. Desaulniers, J. Desrosiers, and M. M. Solomon, Eds., Springer, 2005, ch. 1, pp. 1–32 (cit. on pp. 14, 51).

[40] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," English, *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006, ISSN: 0025-5610. DOI: 10.1007/s10107-004-0559-y (cit. on pp. 14, 32).

[41] J. Lundgren, M. Rönnqvist, and P. Värbrand, *Optimization*. Studentlitteratur, 2010, ISBN 978-91-44-04308-0 (cit. on p. 17).

[42] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231–247, 1992 (cit. on p. 18).

[43] R. Dechter, *Constraint Processing*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003 (cit. on pp. 19–21).

[44] K. Apt, *Principles of Constraint Programming*. Cambridge University Press, 2003. DOI: 10.1017/CBO9780511615320 (cit. on pp. 19, 21).

[45] F. Rossi, P. v. Beek, and T. Walsh, *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. USA: Elsevier Science Inc., 2006, ISBN: 0444527265 (cit. on p. 20).

[46] J. N. Hooker, *Integrated Methods for Optimization (International Series in Operations Research & Management Science)*. Berlin, Heidelberg: Springer-Verlag, 2006, ISBN: 0387382720 (cit. on p. 20).

[47] L. de Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, C. R. Ramakrishnan and J. Rehof, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 337–340 (cit. on p. 21).

[48] D. Kroening and O. Strichman, *Decision Procedures - An Algorithmic Point of View, Second Edition*, ser. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2016 (cit. on p. 21).

[49] Paryanto, M. Brossog, M. Bornschlegl, and J. Franke, "Reducing the energy consumption of industrial robots in manufacturing systems," English, *The International Journal of Advanced Manufacturing Technology*, pp. 1–14, 2015, ISSN: 0268-3768. DOI: 10.1007/s00170-014-6737-z (cit. on p. 27).

[50] D. Meike, M. Pellicciari, and G. Berselli, "Energy efficient use of multirobot production lines in the automotive industry: Detailed system modeling and optimization," *Automation Science and Engineering, IEEE Transactions on*, vol. 11, no. 3, pp. 798–809, Jul. 2014, ISSN: 1545-5955. DOI: 10.1109/TASE.2013.2285813 (cit. on p. 27).

[51] R. Saidur, "A review on electrical motors energy use and energy savings," *Renewable and Sustainable Energy Reviews*, vol. 14, no. 3, pp. 877–898, 2010, ISSN: 1364-0321. DOI: http://dx.doi.org/10.1016/j.rser.2009.10.018 (cit. on p. 27).

[52] O. Maimon, E. Profeta, and S. Singer, "Energy analysis of robot task motions," *Journal of Intelligent and Robotic Systems*, vol. 4, no. 2, pp. 175–198, 1991, ISSN: 0921-0296. DOI: 10.1007/BF00440418 (cit. on p. 27).

[53] M. Ron, P. Burget, and O. Fiala, "Identification of operations at robotic welding lines," in *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*, Aug. 2015, pp. 470–476. DOI: 10.1109/CoASE.2015.7294124 (cit. on p. 27).

[54] S. Bjorkenstam, D. Gleeson, R. Bohlin, J. Carlson, and B. Lennartson, "Energy efficient and collision free motion of industrial robots using optimal control," in *Automation Science and Engineering (CASE), 2013 IEEE International Conference on*, Aug. 2013, pp. 510–515. DOI: 10.1109/CoASE.2013.6654025 (cit. on p. 27).

[55] B. Lennartson, K. Bengtsson, O. Wigström, and S. Riazi, "Modeling and optimization of hybrid systems for the tweeting factory," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 191–205, Jan. 2016, ISSN: 1545-5955 (cit. on p. 29).

[56] K. M. Lynch and F. C. Park, *Modern Robotics: Mechatronics, Planning and Control*, 1st. Cambridge University Press, 2017, ISBN: 9781107156302 (cit. on pp. 29, 30).

[57] C. Hansen, J. Kotlarski, and T. Ortmaier, "Optimal motion planning for energy efficient multi-axis applications," *International Journal of Mechatronics and Automation*, vol. 4, no. 3, pp. 147–160, 2014 (cit. on p. 30).

[58] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st. Springer Publishing Company, Incorporated, 2008, ISBN: 1846286417, 9781846286414 (cit. on p. 30).

[59] D. Verscheure, B. Demeulenaere, J. Swevers, J. D. Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, Oct. 2009, ISSN: 0018-9286. DOI: 10.1109/TAC.2009.2028959 (cit. on p. 31).

[60] Paryanto, M. Brossog, J. Kohl, J. Merhof, S. Spreng, and J. Franke, "Energy consumption and dynamic behavior analysis of a six-axis industrial robot in an assembly system," *Procedia CIRP*, vol. 23, pp. 131–136, 2014, ISSN: 2212-8271 (cit. on p. 31).

[61] K. Paes, W. Dewulf, K. V. Elst, K. Kellens, and P. Slaets, "Energy efficient trajectories for an industrial ABB robot," *Procedia CIRP*, vol. 15, pp. 105–110, 2014, 21st CIRP Conference on Life Cycle Engineering, ISSN: 2212-8271 (cit. on p. 31).

[62] F. Beer, R. Johnston, and P. Cornwell, "Vector mechanics for engineers," in, 9th ed. McGraw-Hill, 2010, ch. 17, p. 1087 (cit. on p. 31).

[63] C. Gaz, F. Flacco, and A. D. Luca, "Identifying the dynamic model used by the KUKA LWR: A reverse engineering approach," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1386–1392. DOI: 10.1109/ICRA.2014.6907033 (cit. on p. 34).

[64] A. Jubien, M. Gautier, and A. Janot, "Dynamic identification of the Kuka LWR robot using motor torques and joint torque sensors data," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 8391–8396, 2014, 19th IFAC World Congress, ISSN: 1474-6670. DOI: http://dx.doi.org/10.3182/20140824-6-ZA-1003.01079 (cit. on p. 34).

[65] A. Othman, K. Belda, and P. Burget, "Physical modelling of energy consumption of industrial articulated robots," in *Control, Automation and Systems (ICCAS), 2015 15th International Conference on*, Oct. 2015, pp. 784–789. DOI: 10.1109/ICCAS.2015.7364727 (cit. on p. 34).

[66] *Experimental packages for KUKA manipulators within ROS-Industrial*, Accessed: 2017-03-17. [Online]. Available: https://github.com/ros-industrial/kuka_experimental (cit. on p. 34).

[67] *Model and simulate multibody mechanical systems*, Accessed: 2017-03-17. [Online]. Available: https://se.mathworks.com/products/simmechanics.html (cit. on p. 34).

[68] *Fast research interface library*, Accessed: 2017-03-17. [Online]. Available: http://cs.stanford.edu/people/tkr/fri/html (cit. on p. 36).

[69] P. Toth and D. Vigo, Eds., *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, English, ser. MOS-SIAM Series on Optimization 18. SIAM, 2014, ISBN: 9781611973587 (cit. on p. 39).

[70] M. Franceschelli, D. Rosa, C. Seatzu, and F. Bullo, "Gossip algorithms for heterogeneous multi-vehicle routing problems," *Nonlinear Analysis: Hybrid Systems*, vol. 10, pp. 156–174, 2013, ISSN: 1751-570X (cit. on pp. 39–41).

[71] J. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, pp. 238–252, 1962 (cit. on p. 42).

[72] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Trans. Netw.*, vol. 14, no. SI, pp. 2508–2530, Jun. 2006 (cit. on p. 52).

[73] É. D. Taillard and S. Voss, *Popmusic — Partial Optimization Metaheuristic under Special Intensification Conditions*, ser. Operations Research/Computer Science Interfaces Series. Boston, MA: Springer, 2002, vol. 15 (cit. on p. 52).

[74] *Service robots: Global sales value reaches 12.9 billion USD*, `https://ifr.org/ifr-press-releases/news/service-robots-global-sales-value-reaches-12.9-billion-usd`, Accessed: 2020-03-16 (cit. on p. 59).

[75] "Automated guided vehicle market size, share and trends analysis report," Grand View Research, Tech. Rep. GVR-1-68038-153-5, 2020. [Online]. Available: `https://www.grandviewresearch.com/industry-analysis/automated-guided-vehicle-agv-market` (cit. on p. 59).

[76] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959 (cit. on p. 62).

[77] W. J. Buchanan, *(CBR) The Handbook of Data Communications and Networks*, 2nd. USA: Kluwer Academic Publishers, 2005, ISBN: 1402077424 (cit. on p. 64).

[78] F. Belik, "An efficient deadlock avoidance technique," *IEEE Transactions on Computers*, vol. 39, no. 7, pp. 882–888, Jul. 1990, ISSN: 2326-3814. DOI: `10.1109/12.55690` (cit. on p. 64).

[79] M. P. Fanti and B. Turchiano, "Deadlock avoidance in automated guided vehicle systems," in *2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings (Cat. No.01TH8556)*, vol. 2, Jul. 2001, 1017–1022 vol.2 (cit. on p. 65).

[80] S. Mohajerani, R. Malik, and M. Fabian, "A framework for compositional synthesis of modular nonblocking supervisors," *IEEE Transactions on Automatic Control*, vol. 59, pp. 150–162, 2014 (cit. on p. 65).

[81] F. Hagebring and B. Lennartson, "Time-optimal control of large-scale systems of systems using compositional optimization," *Journal of Discrete Event Dynamic Systems*, vol. 29, no. 3, pp. 411–443, 2019 (cit. on p. 65).

[82] "Productivity/energy optimisation of trajectories and coordination for cyclic multi-robot systems," *Robotics and Computer-Integrated Manufacturing*, vol. 49, pp. 152–161, 2018 (cit. on p. 73).

[83] S. F. Roselli, K. Bengtsson, and K. Åkesson, "SMT solvers for job-shop scheduling problems: Models comparison and performance evaluation," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, Aug. 2018, pp. 547–552 (cit. on p. 74).